A real-time visible surface algorithm

GARY SCOTT WATKINS

UNIVERSITY OF UTAH

JUNE 1970
UTECH-CSc-70-101
COMPUTER SCIENCE, UNIVERSITY OF UTAH
SALT LAKE CITY, UTAH 84112

A REAL TIME VISIBLE SURFACE ALGORITHM

by

Gary Scott Watkins

June 1970 UTECH-CSc-70-101

This research was supported in part by the University of Utah Computer Science Division and the Advanced Research Projects Agency of the Department of Defense, monitored by Rome Air Development Center, Griffiss Air Force Base, New York 13440, under contract AF30(602)-4277. ARPA Order No. 829.

ACKNOWLEDGMENTS

I am thankful to Dr. David C. Evans and Dr. Ivan E. Sutherland for the great amount of time and suggestions each of them gave in the development of this algorithm, and for their help given in preparing this dissertation.

Mr. Roy A. Keir has read many drafts of this Dissertation. It has been his consultation that has assisted in making this final dissertation readable and understandable.

To Mr. Michael Archuleta, I am grateful for his capability and help in creating a program for testing the hidden line algorithm.

I am thankful to Dr. Gordon Romney for introducing me to computer graphics, and for the creative times we worked together.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

ABSTRACT

With the increasing use of computer graphics, a need is growing for a processor capable of displaying solid objects. Environmental simulation and architectural modeling are only two areas that would benefit from such a display processor.

This dissertation describes an algorithm designed for such a processor, and a program for simulating the hardware processor. The hardware processor would be capable of generating pictures of fairly complicated objects at thirty frames per second. Statistics describing its simulated performance have been extracted and are reported within the dissertation.

CHAPTER I

INTRODUCTION

With the introduction of line-drawing displays, it was soon realized that displaying too much information detracted from the meaning and actually confused the picture. For instance, a single cube can create an optical illusion as shown in Figure 1. However, the optical illusion is removed if lines hidden by surfaces in front of them are not displayed (see Figure 2). A different approach could be taken. Instead of determining the hidden lines, an algorithm could find, color, and shade visible surfaces, thus presenting a more true to life picture. For the past several years, different algorithms have been developed for solving the hidden line or visible surface problem. The various algorithms can be classified into several groups.

A. Path of Edges Algorithms

Some solutions to the problem have been found by various methods of tracing along the edges of objects and noting which of the edges are wholly or partially visible. The resulting picture is then the display of the visible segments of edges. Algorithms based on this method have been developed by Roberts [1], Loutrel [2], and Appel [3]. This type of approach does not take into account the resolution of the display but solves the hidden line problem to

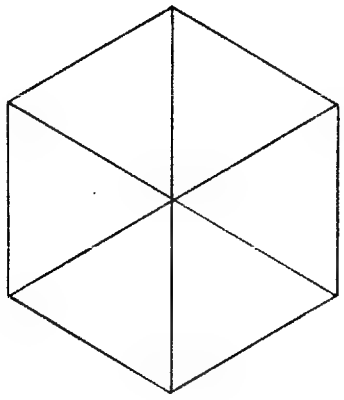


Figure 1

Cube Presenting Optical Illusion

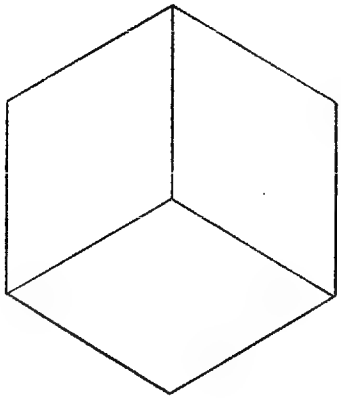


Figure 2

Cube with Hidden Edges not Drawn

the precision inherent in the object description.

B. Sample Space Algorithms

In 1967 a paper was presented by Wylie, Romney, Evans, and Erdahl [4]. One of the concepts discussed was initiated by Evans and introduced the concept of a sample space. The concept states that given an output device with resolution of R_x by R_y, one need only solve the hidden line problem at the discrete resolution points. The sample space can be thought of as taking the original object description in X, Y, Z, and mapping the object on to a two dimensional grid of resolution R_x by R_y. Of course, the Z information needs to be preserved in some form. When this is done, the object will exist only at discrete points in X and Y. The reasoning behind this was when a person views a picture he is physically limited by the resolution of the eye and the resolution of the display device. Hence, the hidden line problem need only be solved to the coarser resolution of the two.

In the algorithm, non-intersecting triangles were used as the object description. However, convex polygons could have been used with only small changes in the program. The algorithm used a scan line approach. That is, one Y raster line would be completely solved for visible triangles before the program proceeded to the next scan line. A method of sorting vertices of the triangles was developed by Wylie

and Romney so only triangles concerned with the current scan line were considered. On each scan line, triangle depths were compared only where edges of the triangles crossed the current scan line. Therefore, it was not necessary to do depth computations at all raster points. Later Romney [5] improved the sorting technique and added a "speedy" check to the program to take advantage of scan line-to-scan line coherence. This improved the speed by eliminating depth sorting as long as triangles entering on the scan line were ordered the same as the previous scan line.

Warnock [6] took a new approach but still kept the grid of resolution points. The object description was generalized by allowing polygons (convex or non-convex) which could intersect one another. Instead of the scan line approach, Warnock took an area of the picture and tried to "understand" it. If it was simple enough to "understand" he would display it, otherwise he subdivided the area into smaller areas. Eventually, a sub-area could be "understood" and displayed, or a sub-area would reach resolution whereupon it would be displayed without further analysis. This concept of subdividing large problems into smaller (and easier) problems is a "non-deterministic" algorithm.

After Warnock's algorithm was developed, Bouknight [7] took the scan line approach and generalized it to include general polygons which could intersect. Figure 3 shows a classification of the various algorithms.

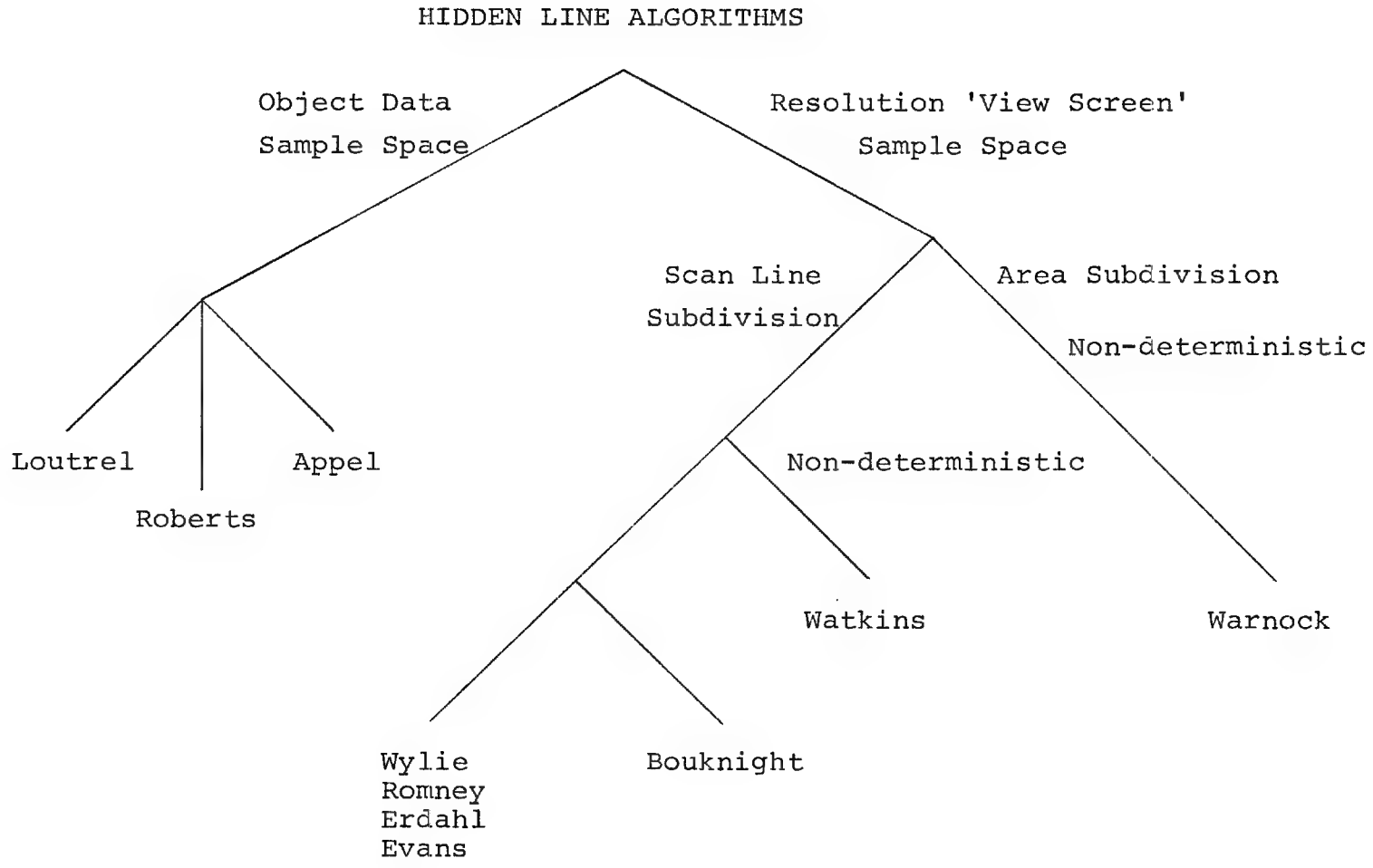


Figure 3

Classification of Algorithms

The new algorithm to be described is of the mapped sample space class, and it allows general polygons which can intersect. Key ideas used in this algorithm are: (1) Scan line-to-scan line coherence of pictures, and (2) an arithmetic unit for Z-depth sorting. Frame-to-frame coherence was not found valuable (in terms of increasing the speed of the program) for inclusion in this final algorithm.

The program implementing this algorithm is a simulation of hardware to generate visible segments of polygons on each scan line at real time speeds. Thus, the program is a Visible Segment Generator (VSG). The output of the VSG is given to a shader for displaying. The method of shading is very similar to that described by Romney [5] and Warnock [6].

CHAPTER II

PRE-FRAME PROCESSING

Before being accepted by the VSG, the object must be processed so that all translations, rotations, and perspective transformations have been applied. All polygons must be clipped at the boundaries of the viewing sample space. Since the scanning process proceeds from Y=1 to Y=512 (or to the Y-resolution value), the edges must be ordered in a list according to the minimum Y value (Y-min) of each edge. Horizontal edges need not be put in the list since the VSG will reject them. On any scan line the VSG can then immediately find which (if any) edges enter on that particular scan line. For each polygon, three fields are zeroed initially and reserved as sorting fields for the VSG. The formats for the edge block and polygon block are shown in Figure 4. The shading and color information will never be used by the VSG for computations. However, the VSG will pass the information to the shader for displaying if the object is visible.

A user that describes objects as closed polyhedra can double the speed of the processor if edge and polygon blocks are only created for polygons that face the viewer. This process was used on the test objects described in Chapter VII.

EDGE BLOCK

POINTER TO NEXT EDGE BLOCK
POINTER TO POLYGON BLOCK
Y - MAX
Y - MIN
X - BEGIN (ASSOCIATED WITH Y - MIN)
ΔX
Z - BEGIN (ASSOCIATED WITH Y - MIN)
ΔZ

POLYGON BLOCK

POINTER TO INITIAL SEGMENT ON POLYGON
POINTER TO NEXT CHANGING POLYGON
POLYGON ACTIVE BIT
SHADING AND COLORING INFORMATION

Figure 4

Description of Edge and Polygon Blocks

CHAPTER III

VISIBLE SEGMENT GENERATOR

The VSG can be broken into three separate processors: (1) Segment Generator, (2) Segment Eliminator, and (3) Depth Sorter.

A. Segment Generator (SG)

The format for a segment block is shown in Figure 5. A segment is defined as the continuous surface of a polygon which exists between two adjacent edges on a scan line. Thus in Figure 6, on scan line 'a' there are two segments, while on scan line 'b' these two segments of the polygon have merged into one segment. A segment block contains a description of the two bounding edges. The two Y-end values specify the Y scan lines when the edges exit from the picture. The X and Z values are stored along with the ΔZ and ΔX increments for each edge. Thus, when the program proceeds to the next scan line, the X and Z values are updated by adding the increments as in Equation 1.

$$Z \leftarrow Z+\Delta Z \quad ; \quad X \leftarrow X+\Delta X \tag{1}$$

The segment blocks are threaded together by four separate list structures:

1. The X-sort list contains all segments on the current scan line sorted with respect to the left edge of each segment. This list has both forward and backward

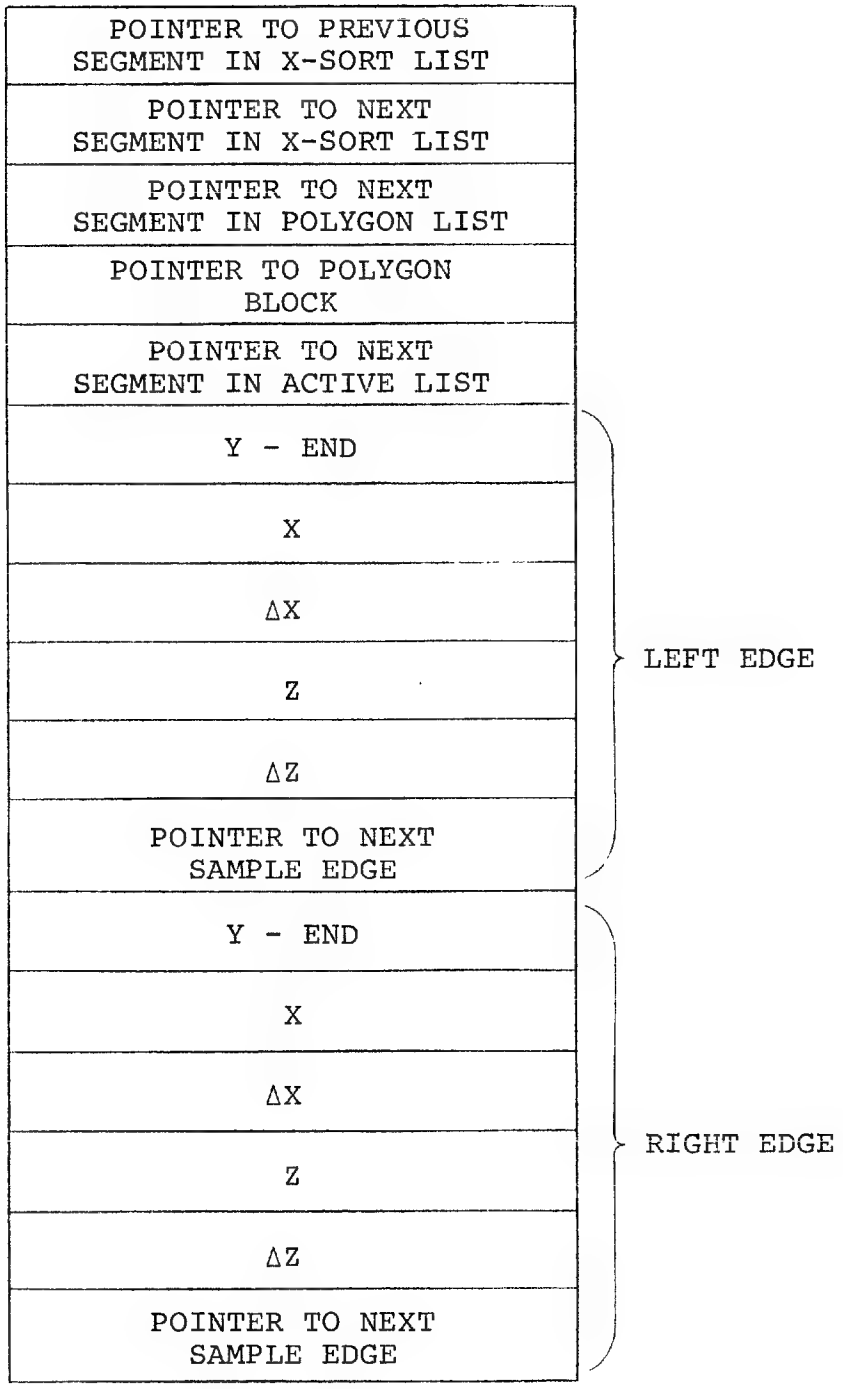


Figure 5

Segment Block

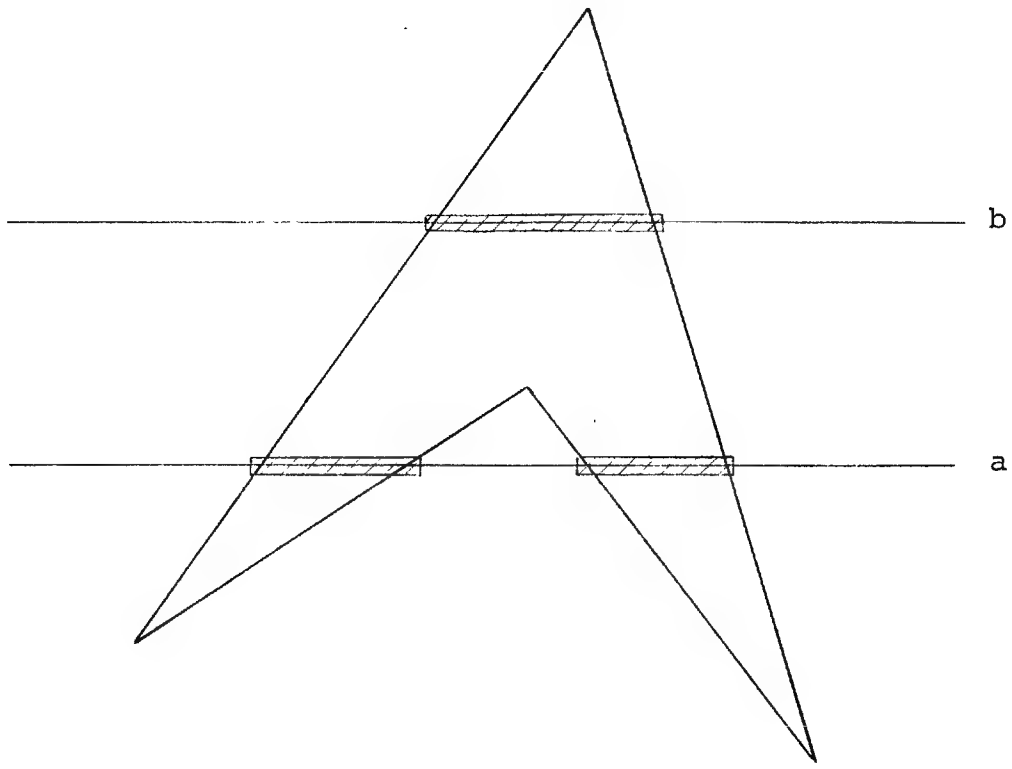


Figure 6
Segments

pointers.

2. Each polygon segments list contains an ordered set of all segments belonging to a particular polygon on a scan line. They are linked together, with the initial pointer (contained in the polygon block) pointing to the left most segment of the polygon.

3. The active segment list contains only segments of the X-sort list which exist in a specified range of X values. Section F of this chapter will give more detail of it.

4. The sample list is another sorted list that will be explained later.

The SG is checked on each scan line to see if any new edges enter the current scan line from the edge list. If there are no entering edges, control is passed to the segment eliminator. If edges do enter on a scan line, data from the edges is used to create a segment.

The polygon block associated with the incoming edge is checked to see if the active bit is set. Active designates whether or not the polygon is already in the list of changing polygons (polygons that have edges entering or exiting on the current scan line). If the polygon was not previously active, it is tagged as active and put in the list containing all changing polygons on this scan line.

Since an edge has only enough data for one half of a segment, an edge can be inserted into either the right or

left side of an empty segment. Because the program does not know whether an edge bounds the right or left side of a polygon, the algorithm may insert an edge into the wrong half of a segment. However, if this happens, the Segment Eliminator will do the necessary rearranging. The X value of the incoming edge is compared against the X values of segments in the polygon segments list until the appropriate location in the list is found for inserting the edge data.

After finding the correct location in the list, and if there is not an empty half of a segment block, the SG must get a block from free storage and insert it in the list at the correct location. Pointers to the segment block must also be inserted in the X-sort list in the correct location whenever data is stored in the left half of the block.

The preceding process is repeated for all edges that enter on the current scan line. Finally when no more edges enter, control is passed to the Segment Eliminator.

B. Segment Eliminator (SE)

The SE runs through the list of all changing polygons, and for each of the polygons it disconnects the polygon from the changing polygon list, and resets the active bit. It then proceeds through the list of segments attached to that polygon to determine if any data needs to be shifted from one segment block to another, or if any segment blocks can be returned to free storage. For example, in Figure 6 on scan line 'a' the polygon has two segments. Because the two

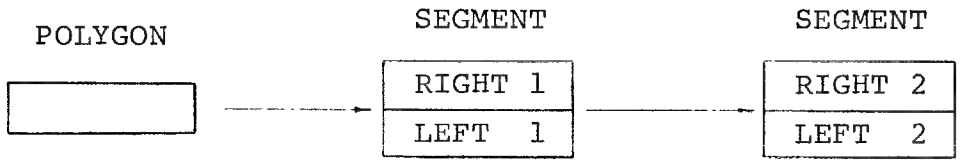
middle edges exit between scan lines 'a' and 'b,' this polygon will have been inserted into the list of changing polygons. The SE must then take the right edge data from the second segment block and insert that data into the right half of the first segment block. After this, the second segment block will be returned to free storage. Figure 7 gives a step-by-step illustration of what would happen if displaying the polygon in Figure 6. When all active polygons have been checked by the SE, control is passed to the Depth Sorter.

C. Depth Sorter (DS)

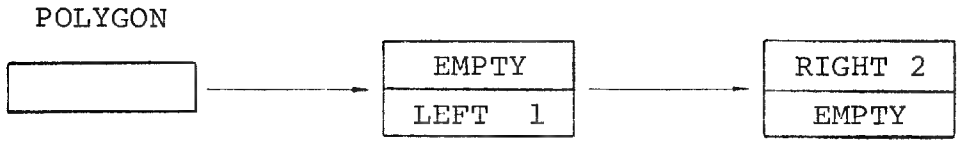
At this point the X-sort list contains all segment blocks on this scan line ordered with respect to the left edge of each scan line. While the SG and SE are concerned only with polygons that change on the current scan line, the DS is concerned with all polygons that exist on the scan line. Therefore, the list handling and memory referencing in this processor are extremely critical to the overall speed of the VSG.

D. Sampling

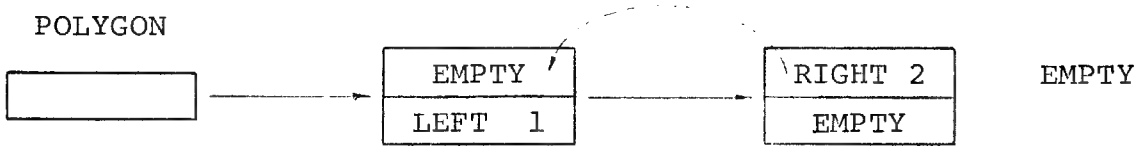
A critical factor in the speed of the algorithm is the number of points on the scan line where depths of polygons are sampled. The depth sorter is capable of determining at most a single visible segment for a restricted span of a scan line. Because of this, sampling must at least be done



Scan Line 'a'



Scan Line 'a' after Noticing Two Edges
Exit on this Scan Line



Scan Line 'b' Showing Data Transfer

Scan Line 'b' after Second Segment Block
Has Been Returned to Free Storage

Figure 7
Packing of Polygon Segment List

at the points of discontinuity (the visible edges). Scan line-to-scan line coherence usually allows the DS to find the visible segment by sampling only at the visible edges contained in the sample list. For the object in Figure 8, one notices the sampling points actually following the visible edges of the picture. Thus the speed of the algorithm will be more dependent on the visible complexity of the object than on the total object complexity.

The Depth Sorter can be subdivided into three separate processors: (1) The Sample Space Generator, (2) The Depth Comparator, and (3) The Decision Processor.

E. Sample Space Generator (SSG)

The SSG operates from the sample list. Essentially the list contains the sorted edges (each half of a segment block is an edge) which were visible on the previous scan line. The building of the Sample List was done on the previous scan line by the Decision Processor and will be discussed under that heading.

The left and right sides of the view screen are always implied sample edges. The scan process on a single scan line proceeds from left to right in X. Therefore, the left edge of the view screen becomes the initial left sample point. The X value of the first edge in the sample list then becomes the right sample point. This sample edge is then removed from the sample list. The portion on the scan line which exists between the left and right sample points

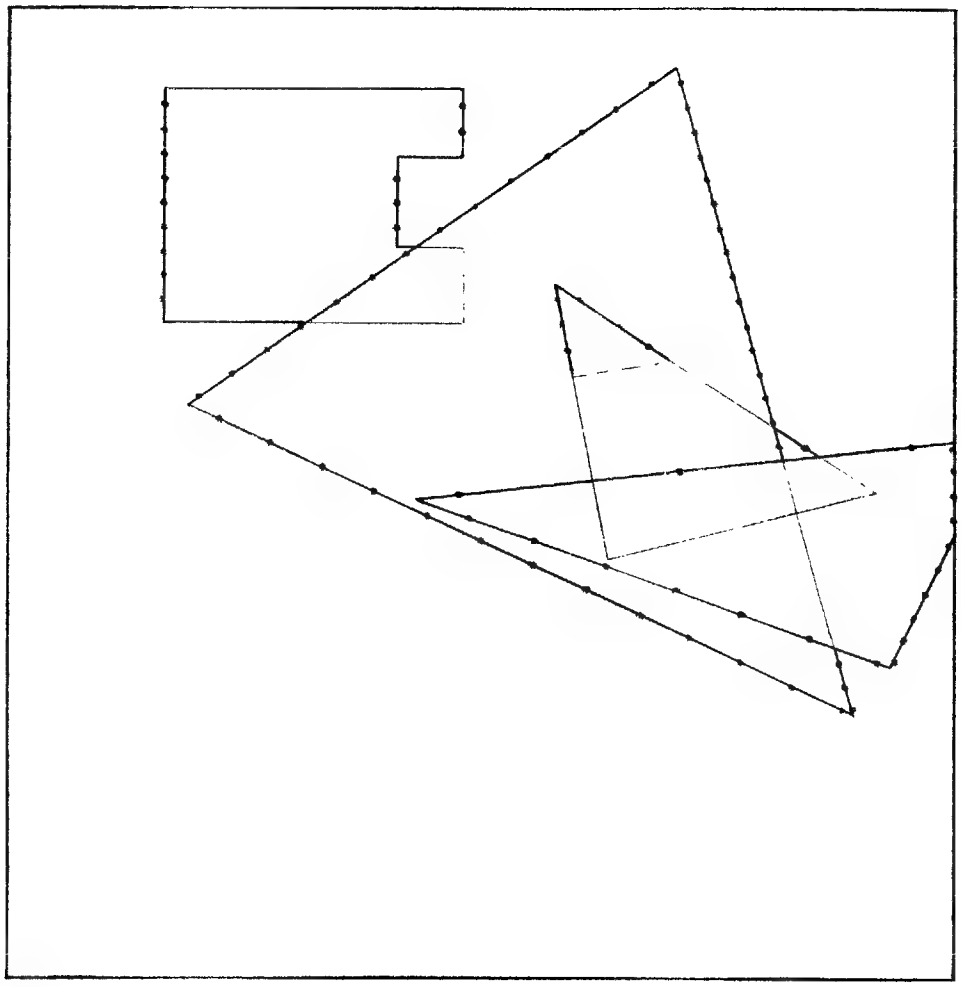


Figure 8
Sampling Points

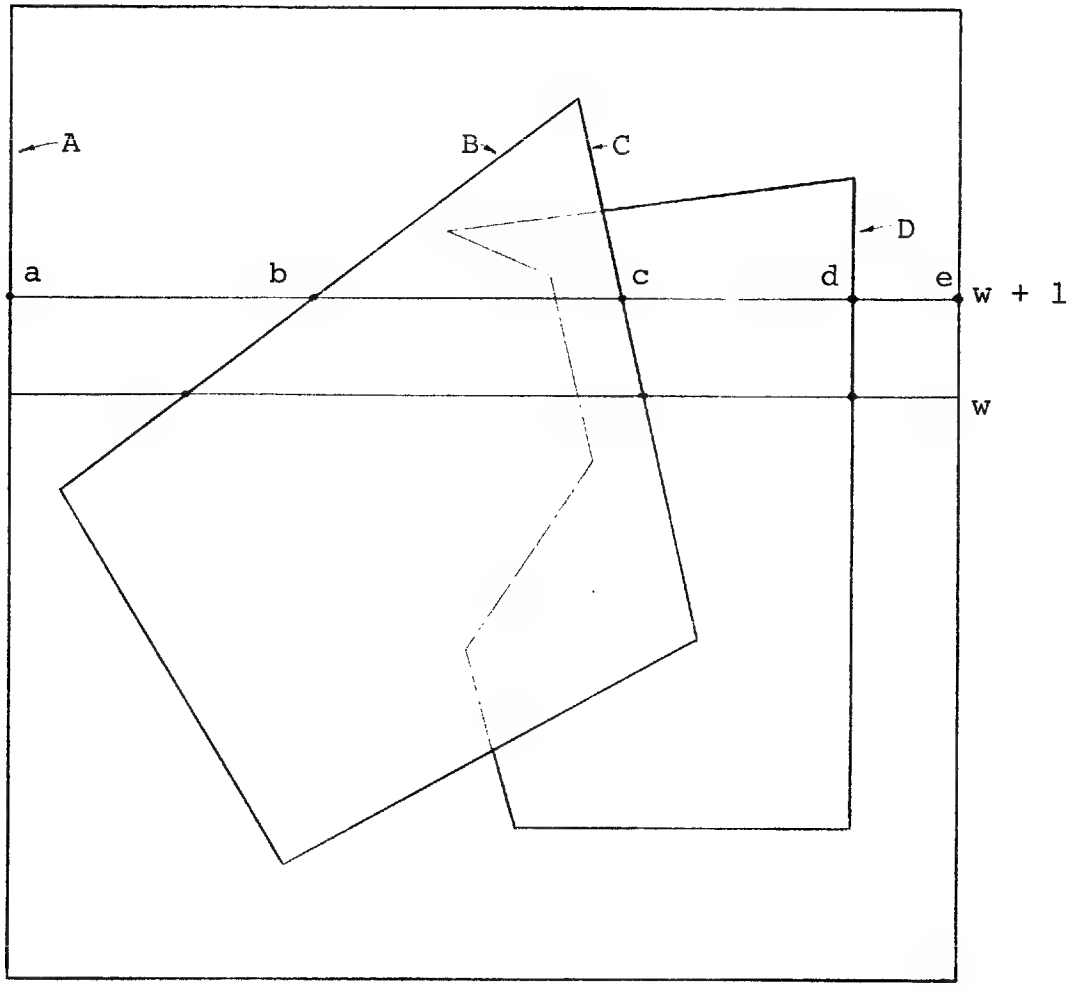
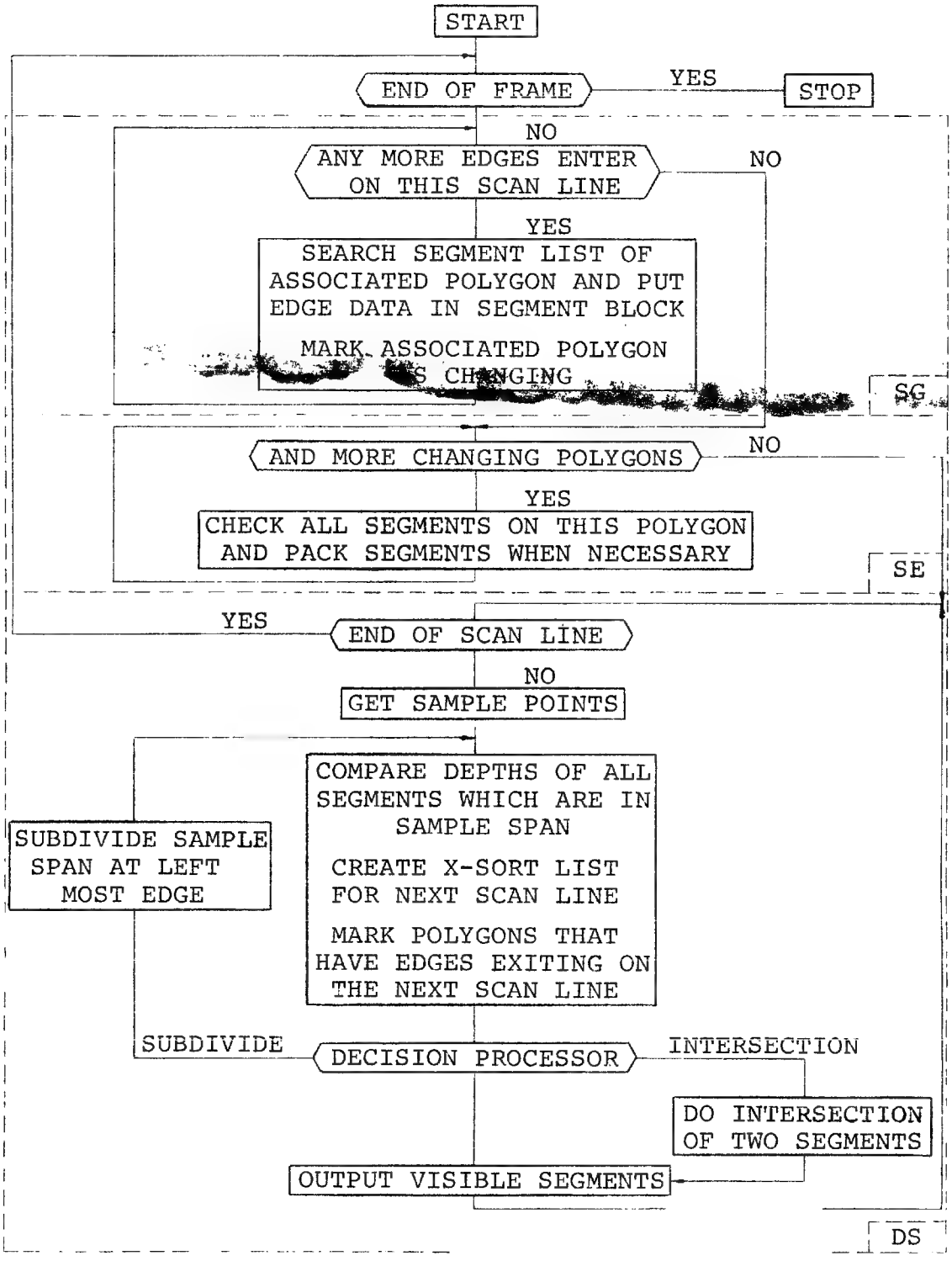


Figure 9
Sample Edges and Sample Points


START
END OF FRAME
YES
STOP
NO
ANY MORE EDGES ENTER ON THIS SCAN LINE
NO
YES
SEARCH SEGMENT LIST OF ASSOCIATED POLYGON AND PUT EDGE DATA IN SEGMENT BLOCK
MARK ASSOCIATED POLYGON
S CHANGING
SG
AND MORE CHANGING POLYGONS
NO
YES
CHECK ALL SEGMENTS ON THIS POLYGON AND PACK SEGMENTS WHEN NECESSARY
SE
YES
END OF SCAN LINE
NO
GET SAMPLE POINTS
COMPARE DEPTHS OF ALL SEGMENTS WHICH ARE IN SAMPLE SPAN
CREATE X-SORT LIST FOR NEXT SCAN LINE
MARK POLYGONS THAT HAVE EDGES EXITING ON THE NEXT SCAN LINE
SUBDIVIDE SAMPLE SPAN AT LEFT MOST EDGE
SUBDIVIDE
DECISION PROCESSOR
INTERSECTION
DO INTERSECTION OF TWO SEGMENTS
OUTPUT VISIBLE SEGMENTS
DS

last segment stored in the X-sort list being prepared for the next scan line. If the new segment X value is larger, it is inserted at the end of the list. If it is not larger, the backpointers of the X-sort list are used until the correct location in the list is found. The surprising data is that line-to-line coherence of the ten test objects (Chapter VII) causes 97 to 99 percent of all segments to be inserted at the end of the list. This means that the X-sort list can always remain sorted in X with very little time spent for rearranging segments. (4) Along with the sorting just discussed, the DC must compare the incoming segment against the currently visible segment. If the incoming segment is in front, it will become the currently visible segment. Every time a new sample span is generated, the first incoming segment becomes the currently visible segment.

If the right edge of a segment extends to the right of the right sample point, the segment must be saved for future depth comparisons when the sample span is moved along the scan line. For this purpose the active segments list was created. Segments are put in the list from the X-sort list and remain only as long as the right edge of the segment is to the right of the left sample point. Therefore, in addition to segments read from the X-sort list, the DC also compares depths of segments read from the active list.

G. Segment Clipping

When two segments are being compared, a clipping algorithm is applied to each of the two segments simultaneously. Figure 11 illustrates the procedure. The two lines represent the segment values on the current scan line. As the Z values of a segment decrease, the segment becomes closer to the observer. Two X clipping values must be obtained. X_{lclip} is defined as the right most left edge in the sample span, and X_{rclip} as the left most right edge in the sample span. If a left edge does not lie in the sample span, the left sample span value is taken as X_{lclip}. In Figure 11, the X value of 'e' becomes X_{lclip} and the X value of 'b' becomes X_{rclip}. A set of registers is then chosen for the left and right clip points of both lines and loaded as in Figure 12.

Since Z_{max} and Z_{min} are stored (not Z_{left} and Z_{right}), an additional bit must be kept which is the sign of $(Z_{left}-Z_{right})$. This bit is used to distinguish the relationship of Z_{left} and Z_{right} to Z_{max} and Z_{min}. Figure 13 shows a more complete gating of the registers contained in dotted box #1 of Figure 12. S of Figure 13 is:

$$S=(XA-X_{lclip}+XB-X_{lclip})/2 \tag{2}$$

or

$$S=(XA+XB)/2-X_{lclip} \tag{3}$$

But (XA+XB)/2 is the midpoint (XM) of the line ab.

$$S=XM-X_{lclip} \tag{4}$$

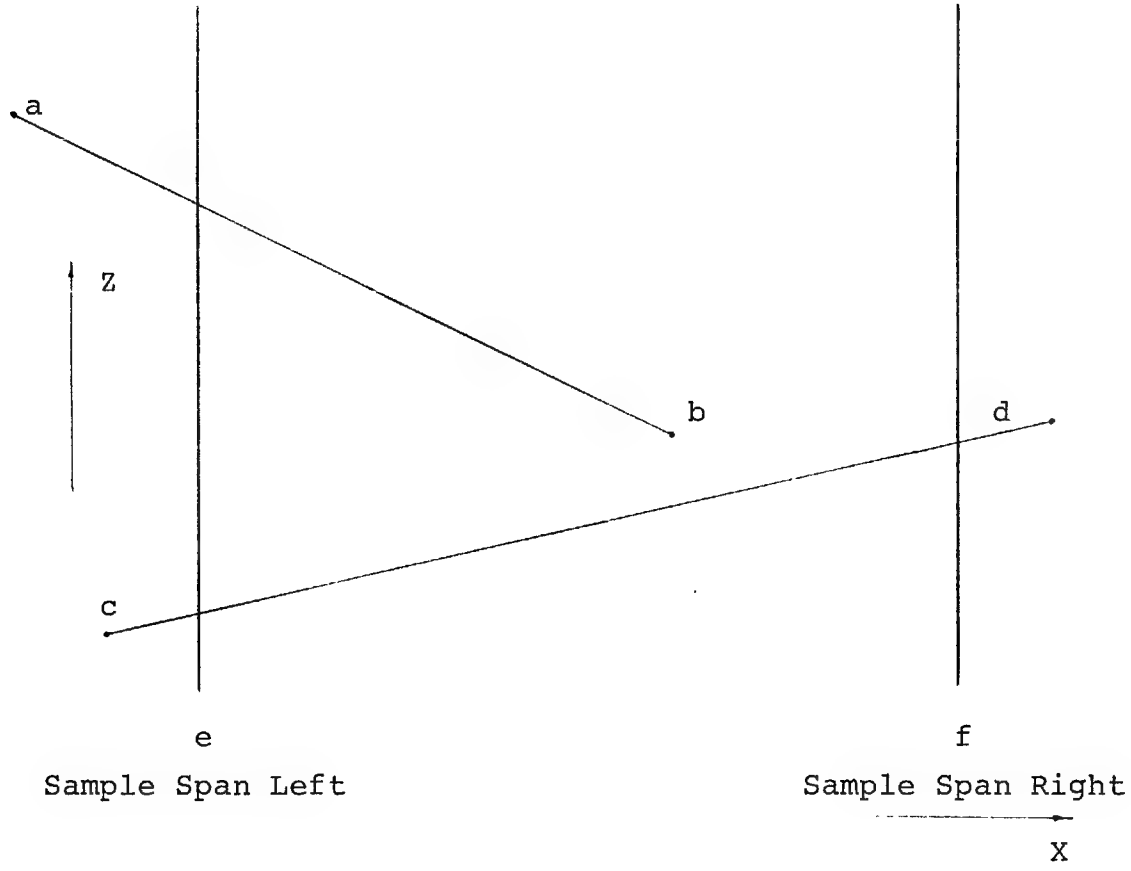


Figure 11
Two Segments on a Scan Line

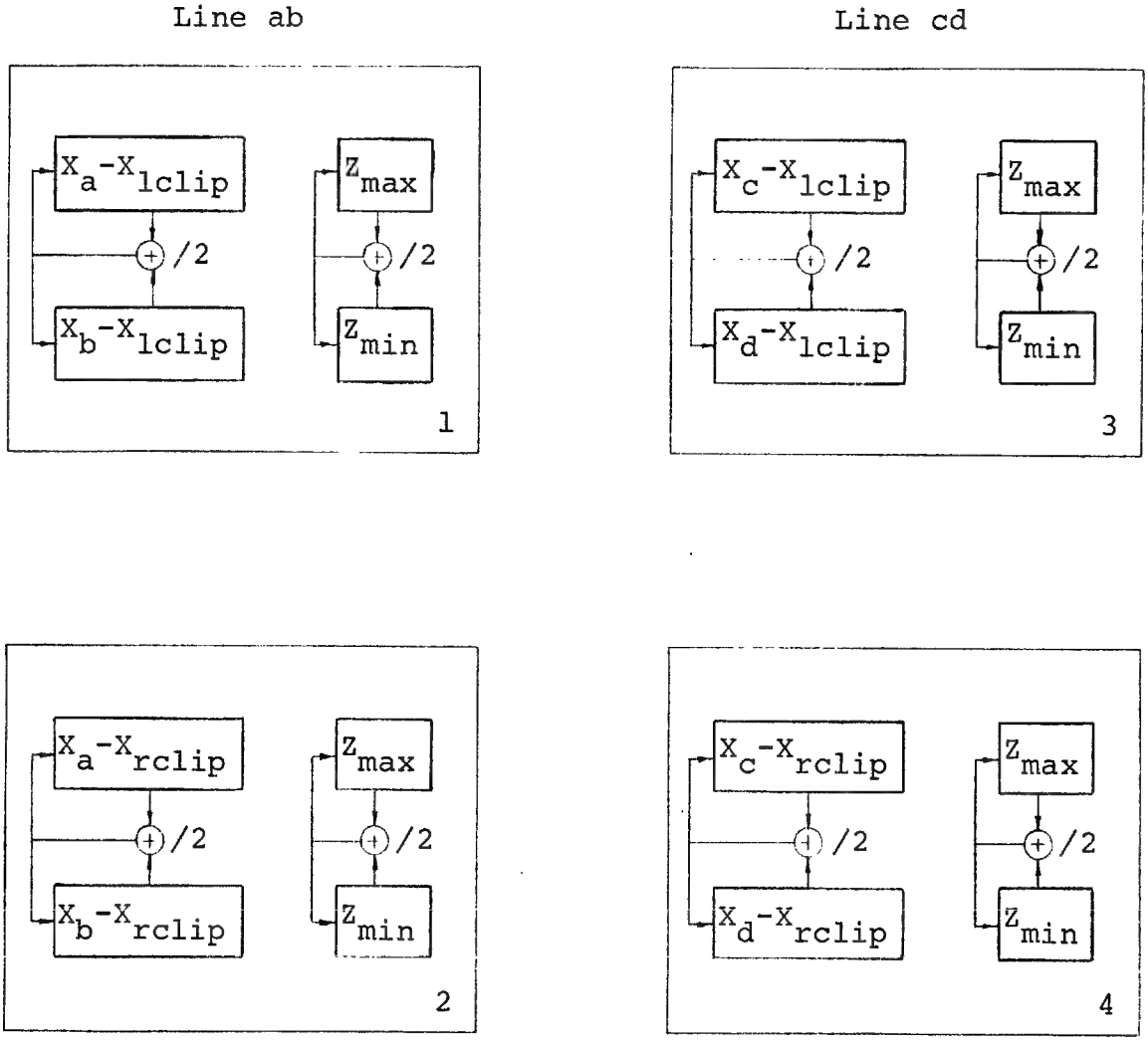
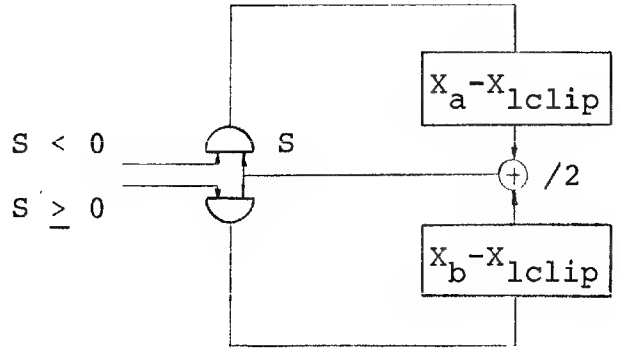


Figure 12

Arithmetic Unit for Depth Comparator



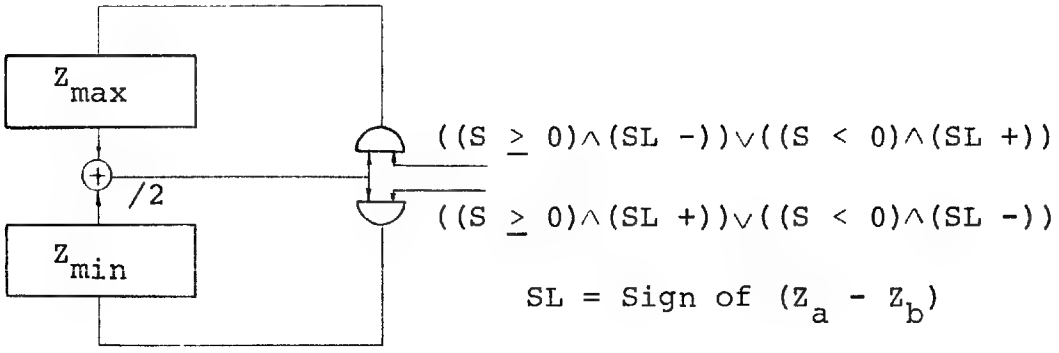
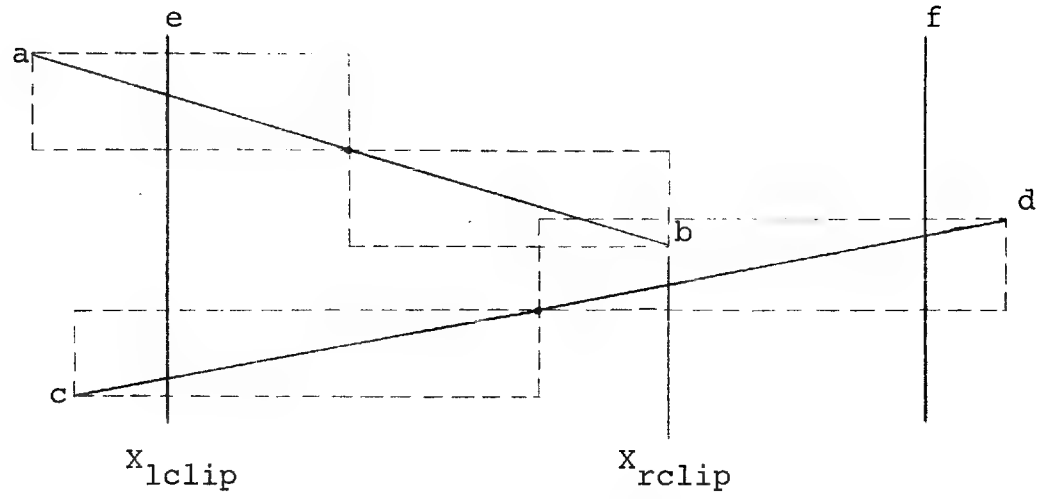


Figure 13
Gating of a Single Quadrant

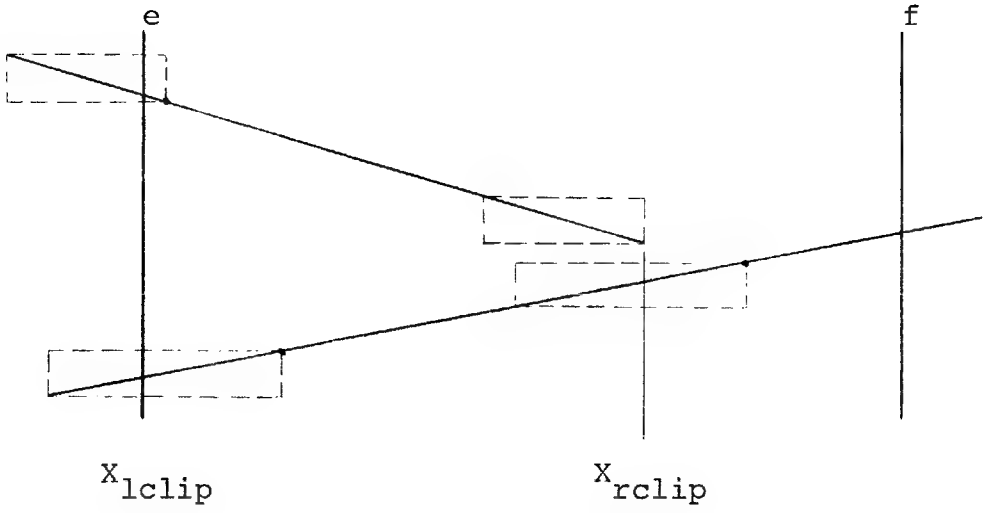
If S is greater or equal to zero, the midpoint is on X_{lclip} or to the right of X_{lclip}. Then the registers containing X and Z of the previous point to the right of X_{lclip} will be replaced with the midpoint of line ab which is closer to X_{lclip}. A similar argument applies if S is less than zero. A more complete description of this clipping process used in a line drawing system is described by Sproull [8].

In Figure 14, succeeding clipping cycles are applied to the two segments of Figure 13. Let Z_{max1} be Z_{max} of quadrant 1 in Figure 12. Z_{min1}, Z_{max2}, Z_{min2}, Z_{max3}, Z_{min3}, Z_{max4} and Z_{min4} are similarly defined. If ($Z_{max1} < Z_{min3}$), line ab is in front of line cd at X_{lclip}. However, as in Figure 14 after one clip cycle, then ($Z_{max3} < Z_{min1}$). Therefore, line cd is in front of line ab at X_{lclip}. Exactly the same argument applies to X_{rclip}, and after two clip cycles line cd is found to be in front of line ab. Since line cd covers line ab everywhere between the sample points 'e' and 'f', it then becomes the currently visible segment.

Many times when lines intersect, or in the case shown in Figure 15, a single currently visible segment cannot be found. In this case a box is made just large enough in X and Z to encompass the two or more lines in question. The amount of data to remember a box description is the same as the amount to remember a line. Also a bit is set declaring a visible box instead of a visible segment. If later a



First Clip Cycle



Second Clip Cycle

Figure 14
Clipping of Segments

segment is found to be in front of the box as in Figure 16, then it becomes the current visible segment and replaces the visible box. The processor continues until all segments that exist in the span are checked. When this is completed, control is passed to the Decision Processor.

H. Decision Processor (DP)

The DP decides whether a visible segment can be put in a display file or if the sample span must be subdivided in some manner and the Depth Comparator started again. If the DP finds there is a visible segment from the DC, it outputs the corresponding segment to the display file. If the DC discovered a visible box, and any of the visible segments in the box have an edge existing within the sample span, the right sample point is set to the X value of that edge (subdivision), and control is passed back to the DP. For instance, the DP would cause the control to subdivide at X=a for segments in Figure 17.

If no edges exist between the left and right sample points, two conditions can exist: (1) For more than two segments existing in the visible box as in Figure 18, the sample span is divided in half. That is, the right sample point is moved half way toward the left sample point. After this subdivision process, control is passed back to the DC again. (2) If only two segments exist in the box, the condition is the intersection case of Figure 19. The

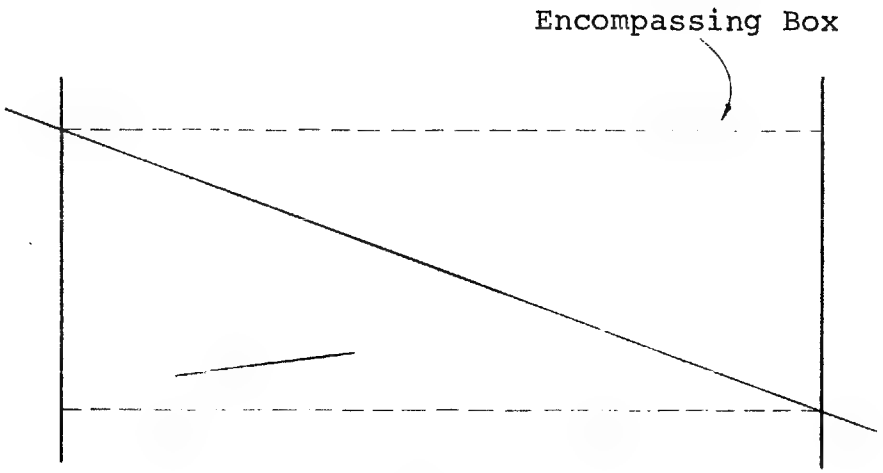


Figure 15
Boxing of Segments

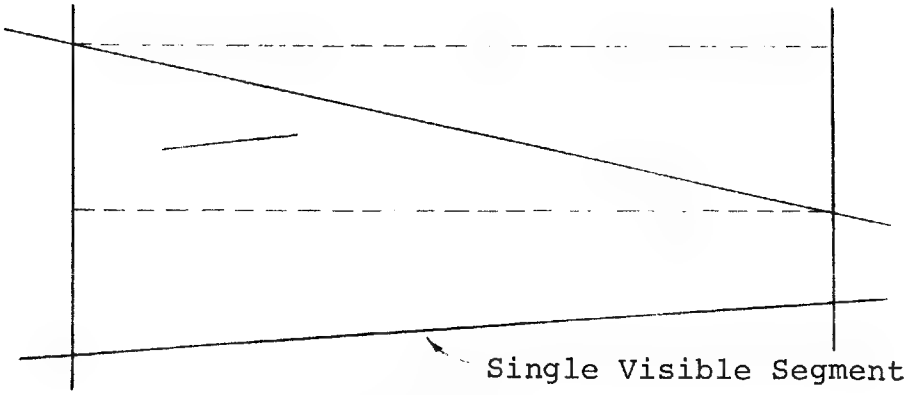


Figure 16
Elimination of Visible Box by Visible Segment

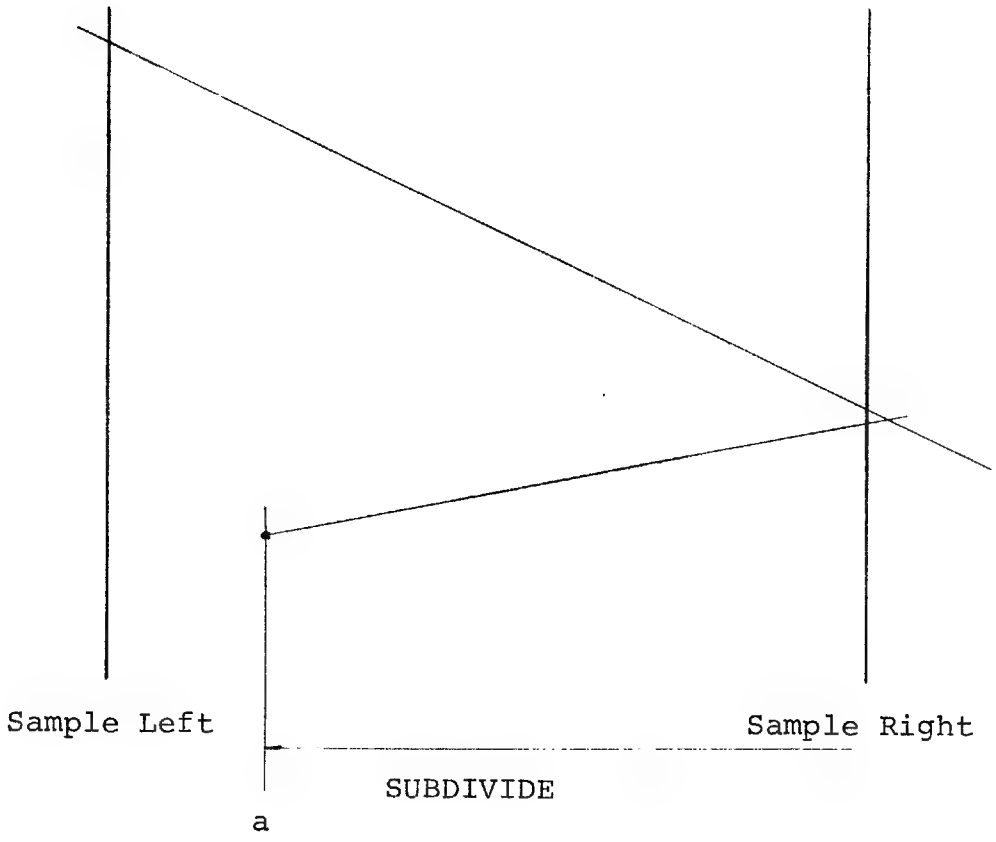


Figure 17
Subdivision

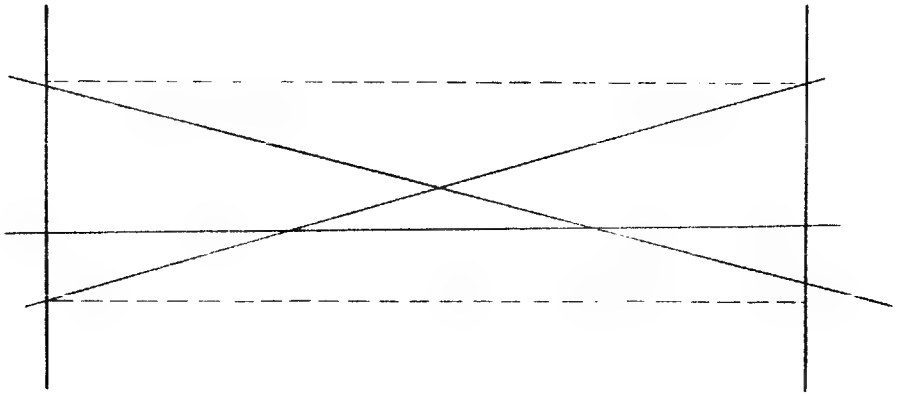


Figure 18
Three Potentially Visible Segments

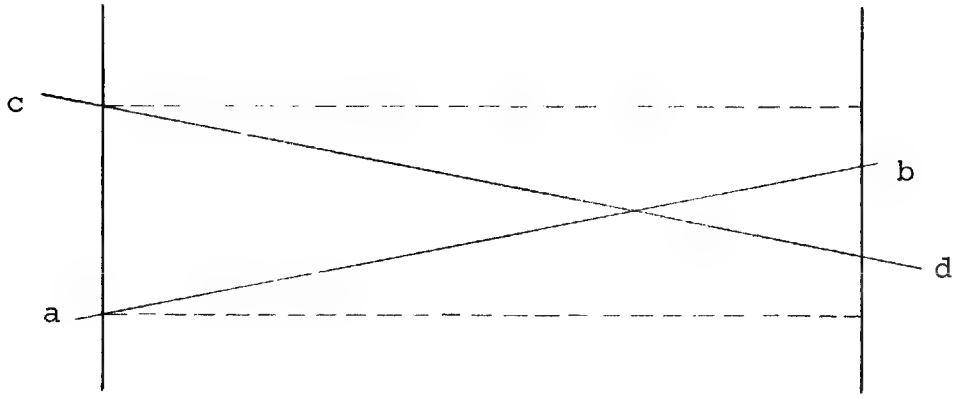


Figure 19
Intersecting Segments

same clipping hardware used for depth comparisons can also be used for calculating the intersection of these two lines.

I. Intersecting Segments

The intersection calculation is done in two stages. First, the registers of Figure 12 are loaded exactly in the same manner as for the DC. However, instead of terminating when the Z_{max} and Z_{min} tests are satisfied, the adders run until all registers contain either 0 or -1. When the registers reach this state, Z_{max1} will hold the Z value of line ab at X_{lclip}, Z_{max2} the Z value of line ab at X_{rclip}, Z_{max3} the Z value of line cd at X_{lclip}, and Z_{max4} the Z value of line cd at X_{rclip}. Figure 19 has been reduced to the problem represented in Figure 20.

For the second stage, the problem can be solved by loading the registers in the manner shown in Figure 21. Because of the intersection, Z_1 and Z_2 will have opposite signs. Therefore, after each add cycle the Z sum is stored into the Z register which has the same sign as the sum. The X registers will also be stored in the same direction determined by the Z sum. After $[\log_2(X_{rclip}-X_{lclip})]$ add times, X_1 and X_2 will both contain the X value of the intersect of the two segments.

A block from free storage is obtained at this point and the X intersect value and the pointers to the two segments causing the intersection are stored as data in an implied edge list. When the program proceeds to the next scan line,

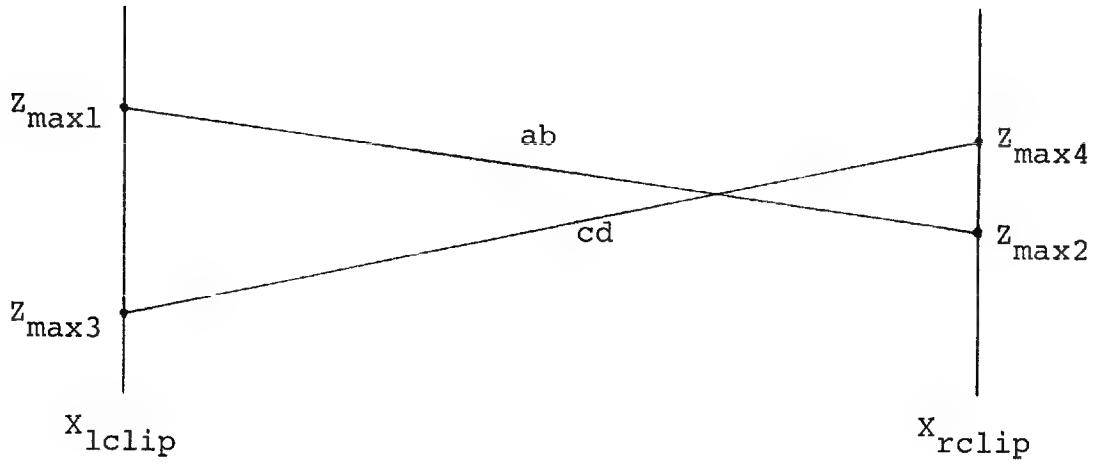


Figure 20
Intersecting Segments Clipped to X_{lclip} and X_{rclip}

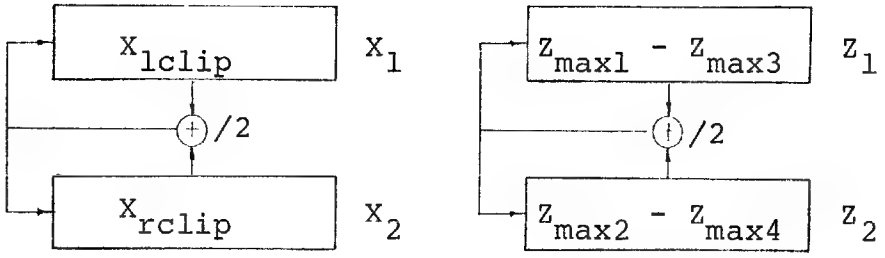


Figure 21
Registers for Finding Intersection

the intersect will again be calculated. The difference between the intersect on this scan line and the intersect calculated on the previous scan line can be used as the increment of the implied edge. This edge can now be treated as any other visible edge and used for determining sample points. If, on a scan line, an implied edge is found to be no longer visible, the block is returned to free storage.

J. Building the Sample List

The DP has one other task. That is, to tag the visible edges (determined in the DP), and put them in the sample list. Upon completion of the DP, control is either passed to the SSG if subdivision did not occur, or to the DC if subdivision did occur.

CHAPTER IV

FRAME-TO-FRAME COHERENCE

This algorithm can easily take advantage of frame-to-frame coherence of pictures. For instance, in a movie if an edge is visible in one frame, it will usually be visible in the next frame. If an edge is found to be visible on the scan line it enters on, the edge block (see Figure 4) is tagged as visible. This means one additional bit must be stored in each edge block. Also two pointers to each edge block must be stored in the segment blocks. Then when the next frame is being processed and an edge was found to be previously visible, the initial X value of the edge is then used as a sample point. The frame-to-frame coherence algorithm was used on some of the earlier versions of the program. However, the scan line-to-scan line coherence was so efficient that the frame-to-frame coherence only decreased the number of memory references by about 0.1 percent. Because of this, it was not implemented in later programs.

CHAPTER V

RELATIONSHIP WITH OTHER ALGORITHMS

On the basis of generality of object descriptions, this new algorithm is as good as or better than the others mentioned in the introduction. Convex or non-convex polygons of any number of sides can be used. The algorithm allows polygons to penetrate one another without any pre-processing checks.

Since planar equations are never used for depth sorting, the algorithm can not tell if the points of the polygons lie on a plane. It always assumes a linear interpolation between the edges on a scan line. However, when shading a polygon a discontinuity in shading can be created. For example, if the vertex between scan lines 'a' and 'b' of Figure 6 were not on the plane described by the other three vertices, the linear depth calculations between edges would show a discontinuity in the shading between the two scan lines. Furthermore, the line of discontinuity would always remain horizontal even if the polygon were rotated. Also, since segments are only checked when edges enter or exit, edges of a single polygon should never cross each other. If they do cross, however, a local error will occur in the picture only where that polygon exists and if that polygon is visible. Consequently, points of a polygon

should lie on a plane. (Points not on a plane can introduce edges that cross).

Like Warnock's algorithm, this new algorithm is also non-deterministic, but on a scan line level. For instance, a sample span on a scan line is assumed to have one covering polygon. If it does not, the sample span is made smaller until finally a span is found which is covered by a single polygon.

Romney used an ordering scheme for taking advantage of scan line-to-scan line coherence. He did not allow intersecting triangles. Therefore, as long as the intersection of the edges of triangles on the current scan line were in the same order as on the previous scan line, the same triangles that were visible previously would be visible on this scan line. However, as soon as the order changed, the remainder of the scan line had to be depth sorted. The coherence ordering made a great difference in the speed of his algorithm.

If intersections are allowed, as in the new algorithm, this ordering of edges no longer holds for determining visibility. Therefore, the sampling process described in Chapter III-D was developed. It has the further advantage that even when the order changes, the previously calculated sample points for the remainder of the scan line are still valid.

CHAPTER VI

DEVELOPMENT OF THE NEW ALGORITHM

As is usually the case in the development of new algorithms, the process was evolutionary. Successive algorithms were developed, tested, and improved upon. The history of this algorithm can be divided into six distinct steps. These programs are called VSG1, VSG2, etc.

1. The first step used edges on each scan line. The edges were sorted in X separately, and after sorting they were read in order. Every time an even number of edges was found associated with a polygon, a segment block was created from free storage. Finally, the segments were depth sorted for visibility.

2. VSG2 linked the edges together with pointers after sorting in X. This eliminated the creation of segment blocks on each scan line.

3. VSG3 took the edge data and created segment blocks only when edges entered on a scan line. These segments are described in Chapter III-F. Since there are one half as many segments as edges, the X-sort on each scan line is twice as fast as in VSG2. Also, edges no longer needed to be linked together on every scan line.

4. VSG4 eliminated the X-sort which was done separately before the depth sorting. The X-sort and depth sort were done simultaneously on each scan line.

5. The four previous algorithms used planar equations and a multiplier for calculating depths of the polygons. A divider was also required for finding the intersect of two polygons. VSG5 replaced the arithmetic unit with the midpoint clipping simulation described in Chapter III-E.

6. Up to this point all algorithms used a bucket sort as described by Romney [5] for sorting segments in X. This final algorithm used the assumption that a sorted list will remain sorted by interchanging only a few segments when proceeding from one scan line to the next.

CHAPTER VII

TEST DATA

Ten objects were chosen to represent various complexities of pictures. Figures 22-31 contain pictures of the objects. Each object has two pictures. One shows all edges in the picture and the other shows the objects after visible surfaces are found and shaded.

A. Objects

Object 1, Penetration: The object is relatively simple but has many intersecting planes.

Object 2, E-S: Many edges abound in the picture and a great amount of visible complexity exists.

Object 3, Low Area: Although intersections abound, the picture only occupies a small area.

Object 4, Cube1: Twenty-five cubes exist, but only the front cube is visible.

Object 5, Cube2: Object 4 has been rotated so that parts of all twenty-five cubes are visible. An enormous amount of visible complexity exists.

Object 6, Shape1: This object is made up of many long and narrow polygons which are long in the X direction.

Object 7, Shape2: Object 6 has been rotated so the polygons are long in the Y direction. These two objects are to show what effect the object orientation can have

on the scanning process.

Object 8, Sheet: This is a wavy object made up of triangles. Everything is at least partly visible.

Object 9, Simple1: This object is made up of a large cube encompassing a sphere and intersecting cubes.

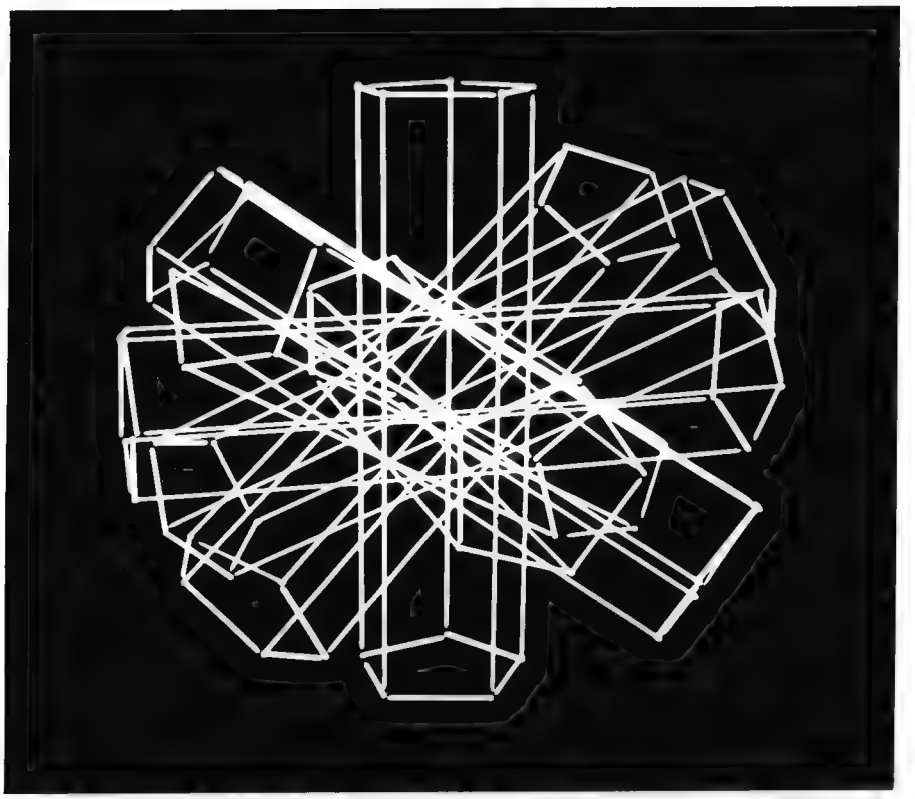
Object 10, Simple2: Object 9 has been changed slightly so the sphere intersects the cube and is partly visible.

B. Statistics

For each of the VSG algorithms mentioned in Chapter VI, statistics were gathered. These statistics included data about the object (number of polygons, etc.), computation required, memory reference counters, and various other counters. Appendix II contains a list of statistics. At the beginning of each set of statistics for a particular algorithm, there is a table describing the various counters. Figure 32 contains a table of statistics that have been extracted for the Penetration object (Figure 22). The statistics of the six various changes in the algorithm are shown for that object. The table in Figure 33 shows a cross section of statistics for all the objects with the final algorithm.

C. Analysis

Before any statistics were gathered, arithmetic computation was suspected to be the bottle-neck in solving the hidden line problem. Statistics, however, showed that



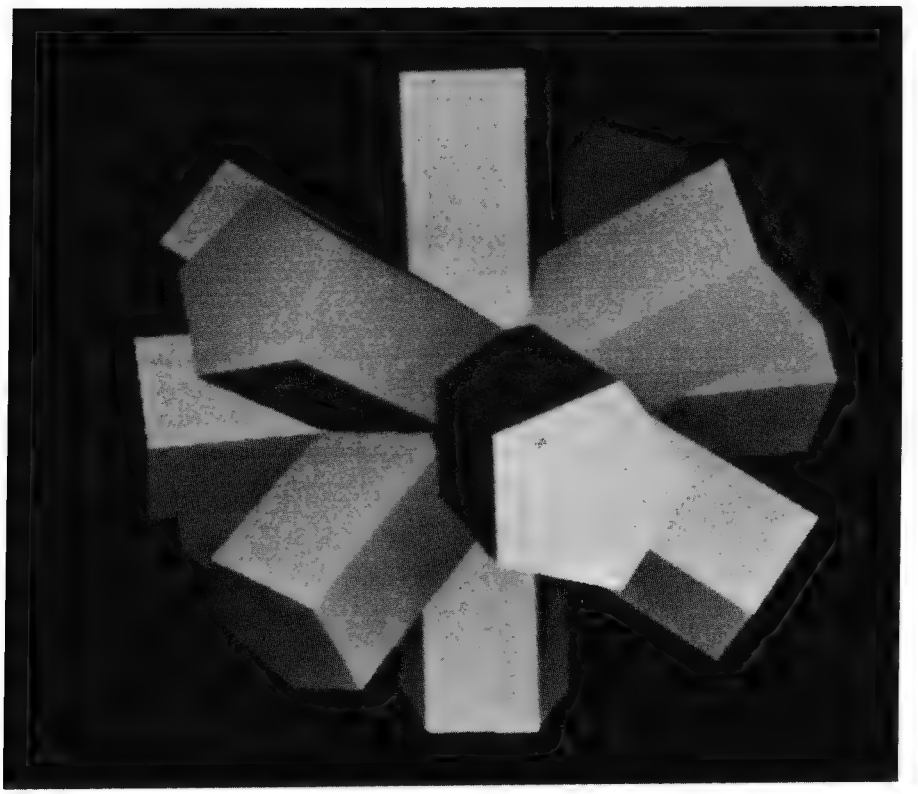


Figure 22
Object 1: Penetration

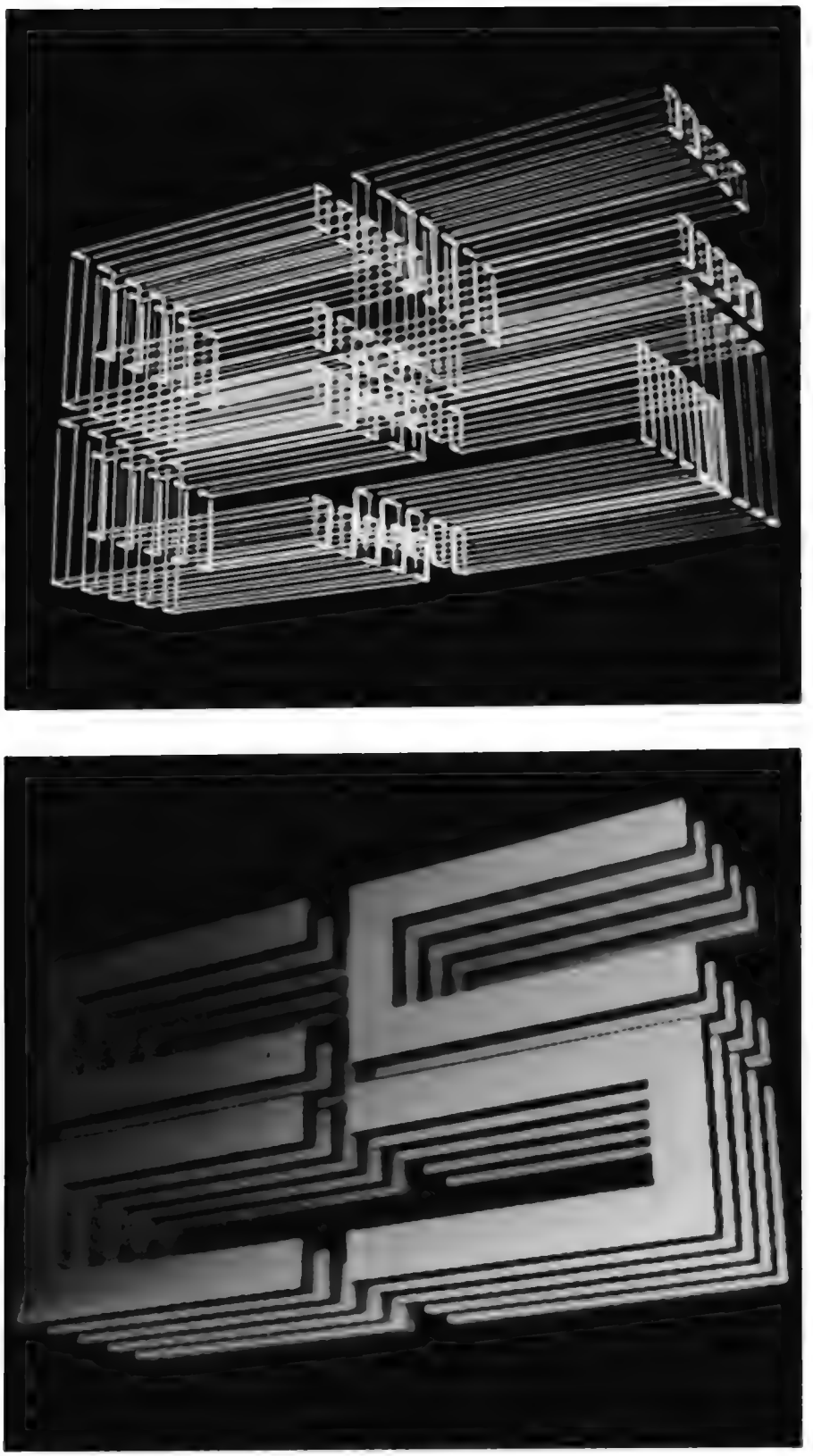


Figure 23
Object 2: E-S

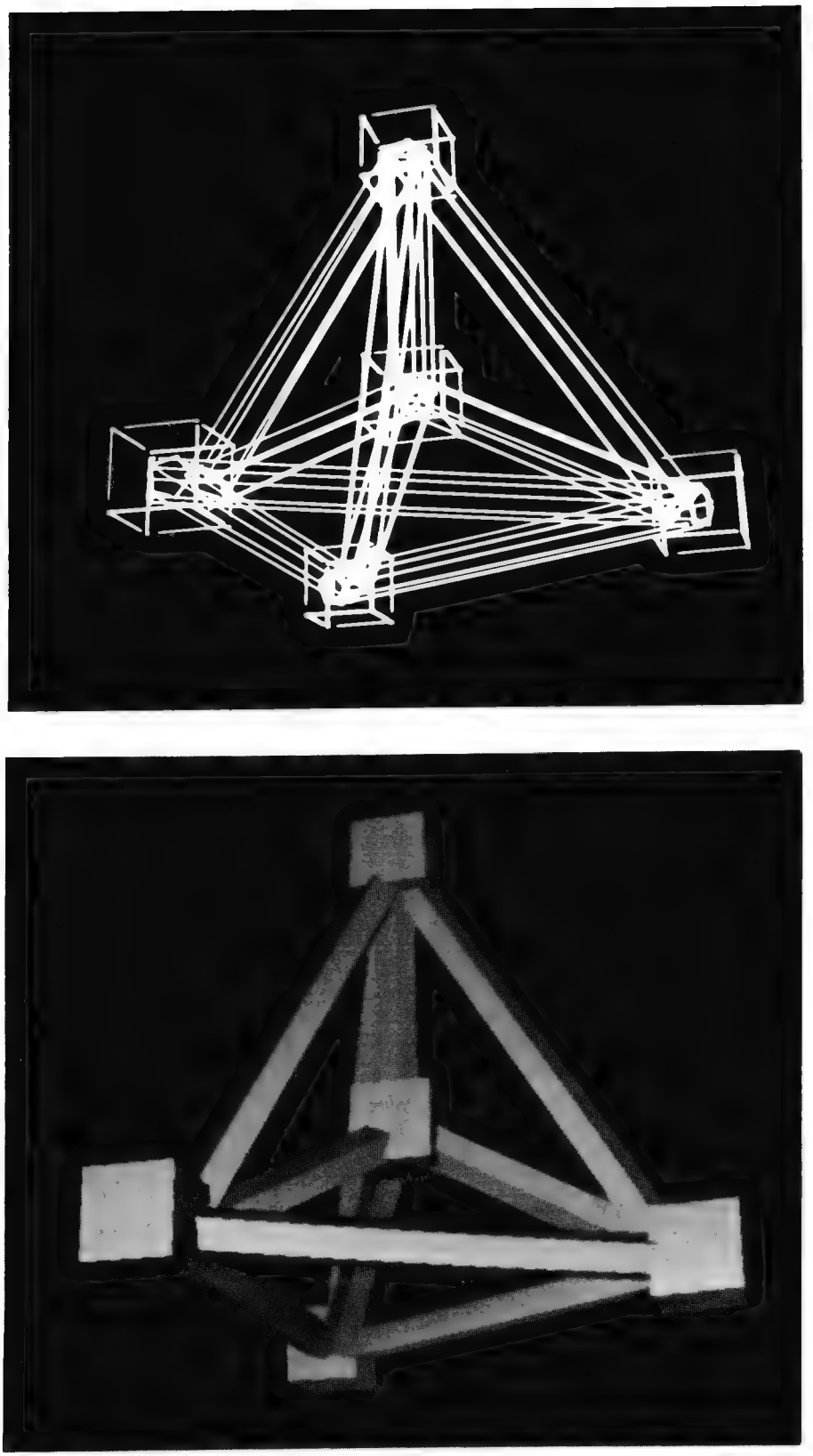


Figure 24
Object 3: Low Area

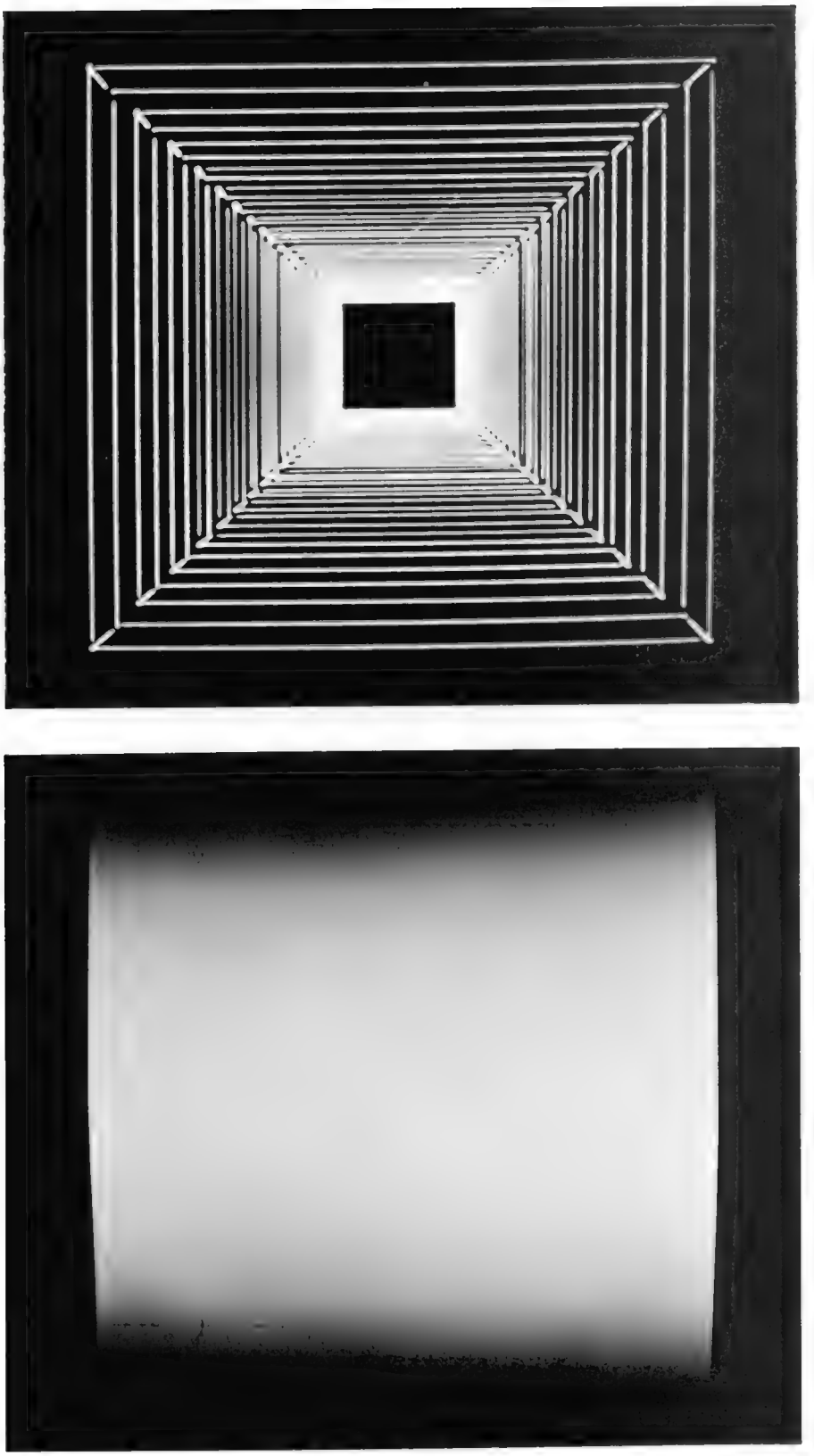


Figure 25
Object 4: Cube1

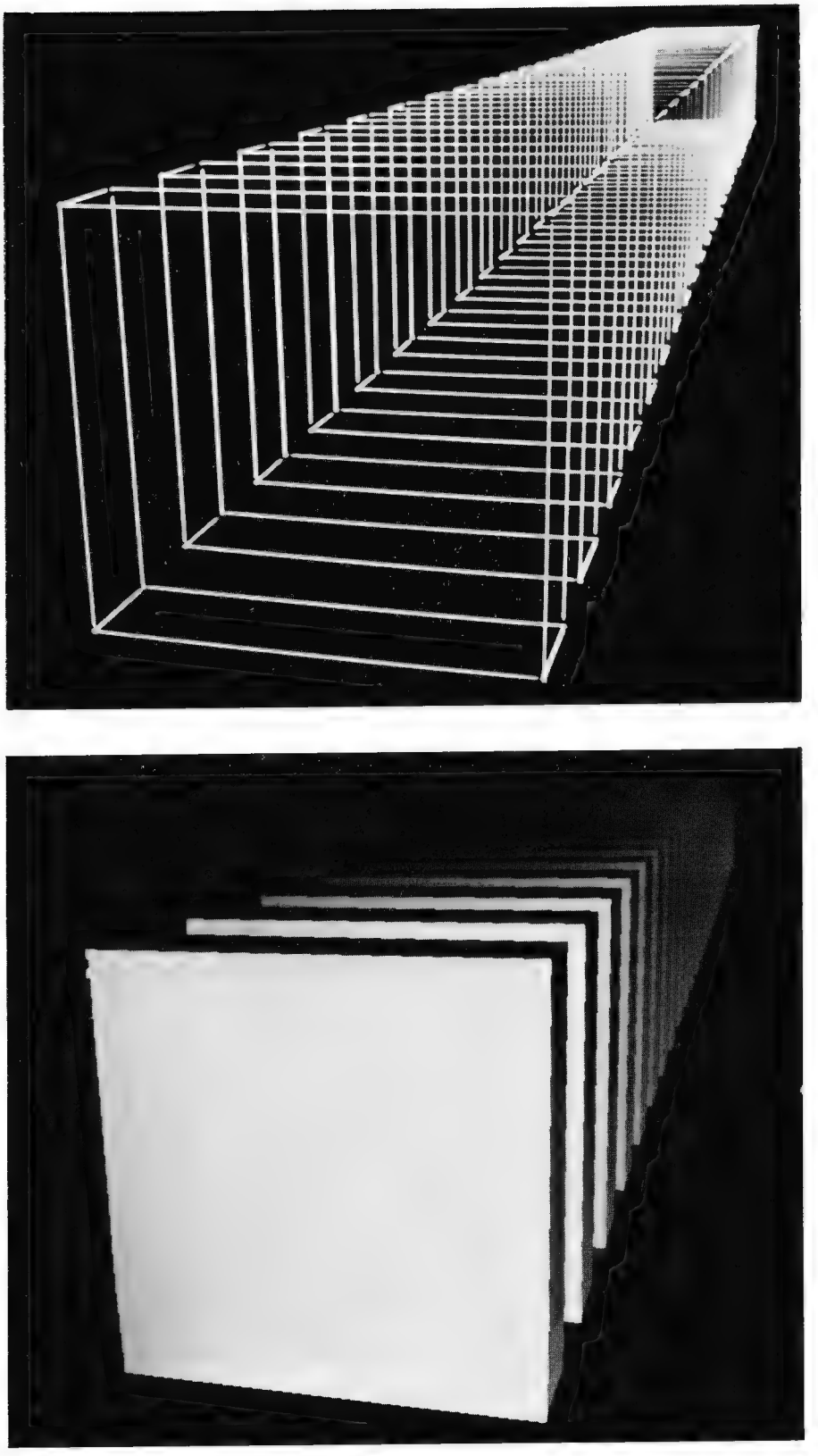


Figure 26
Object 5: Cube2

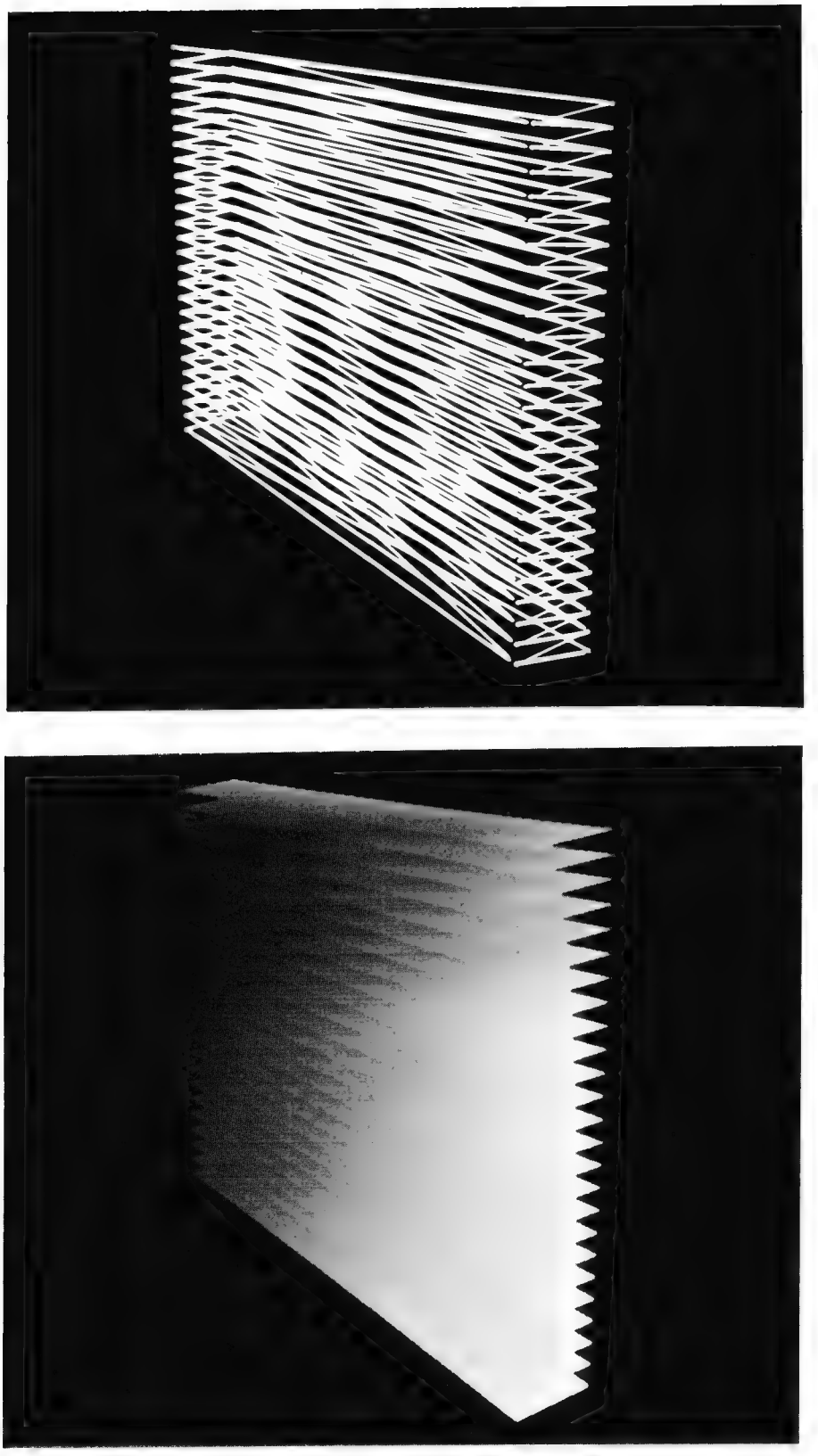


Figure 27
Object 6: Shape1

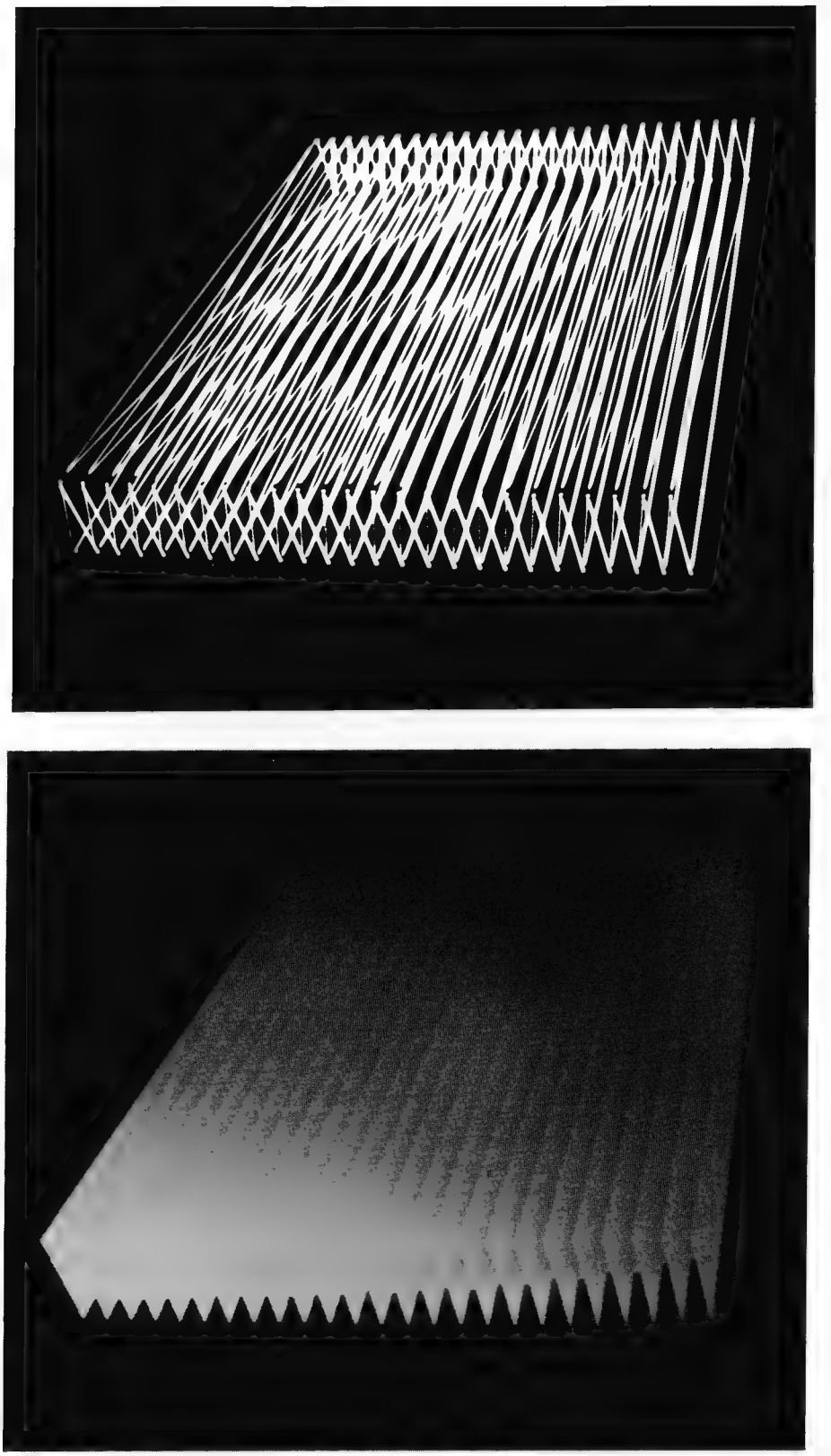


Figure 28
Object 7: Shape2

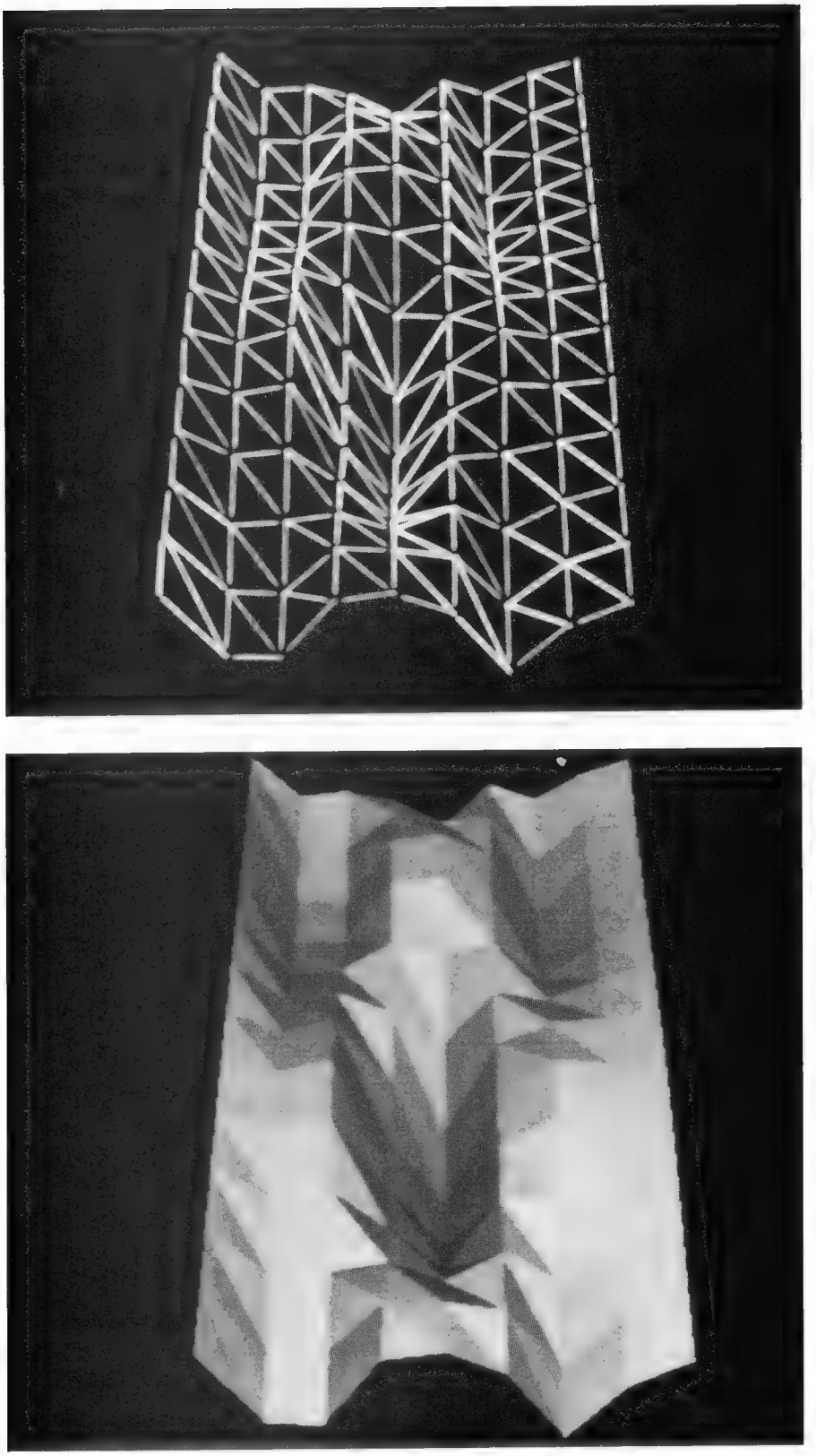


Figure 29
Object 8: Sheet

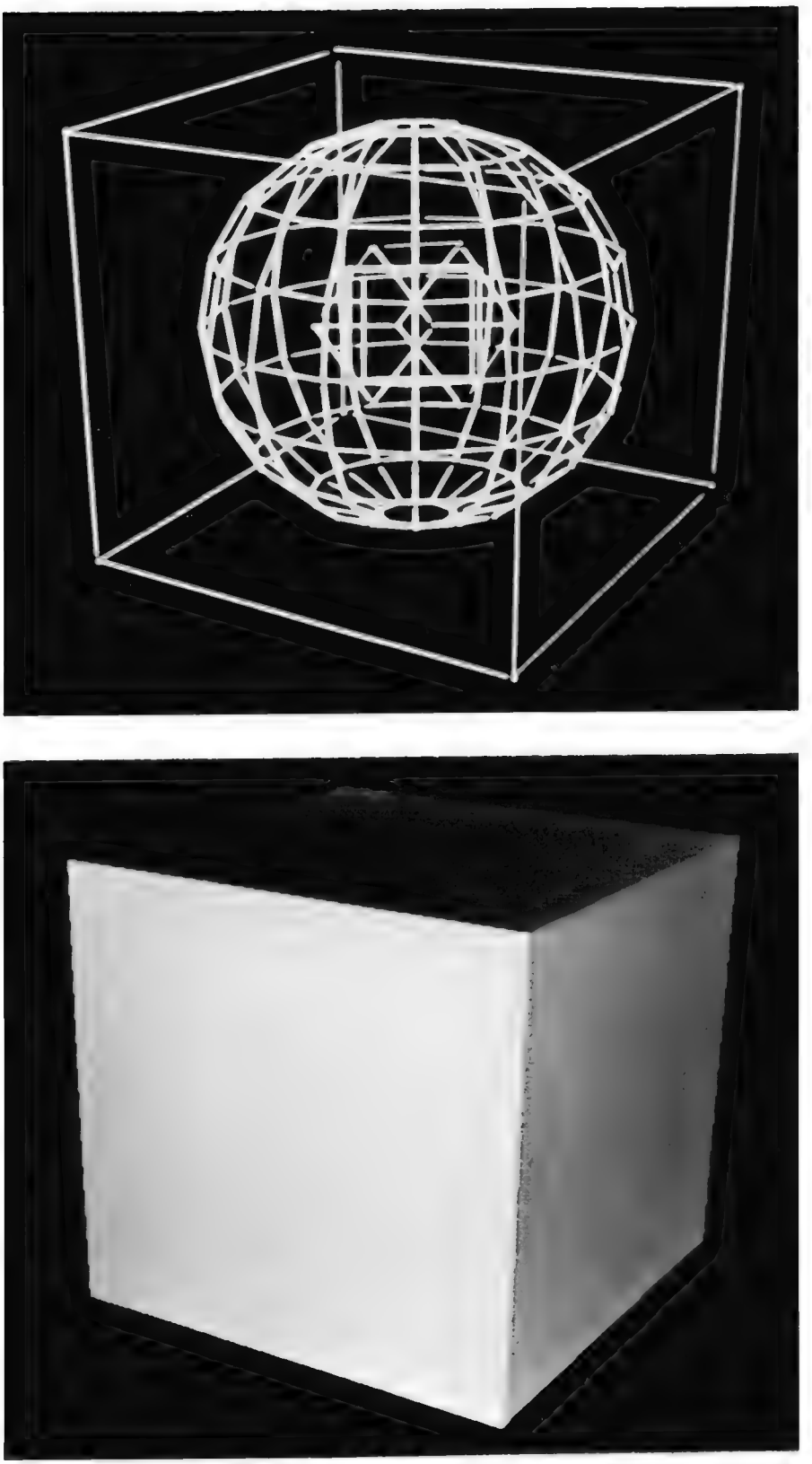


Figure 30
Object 9: Simple1

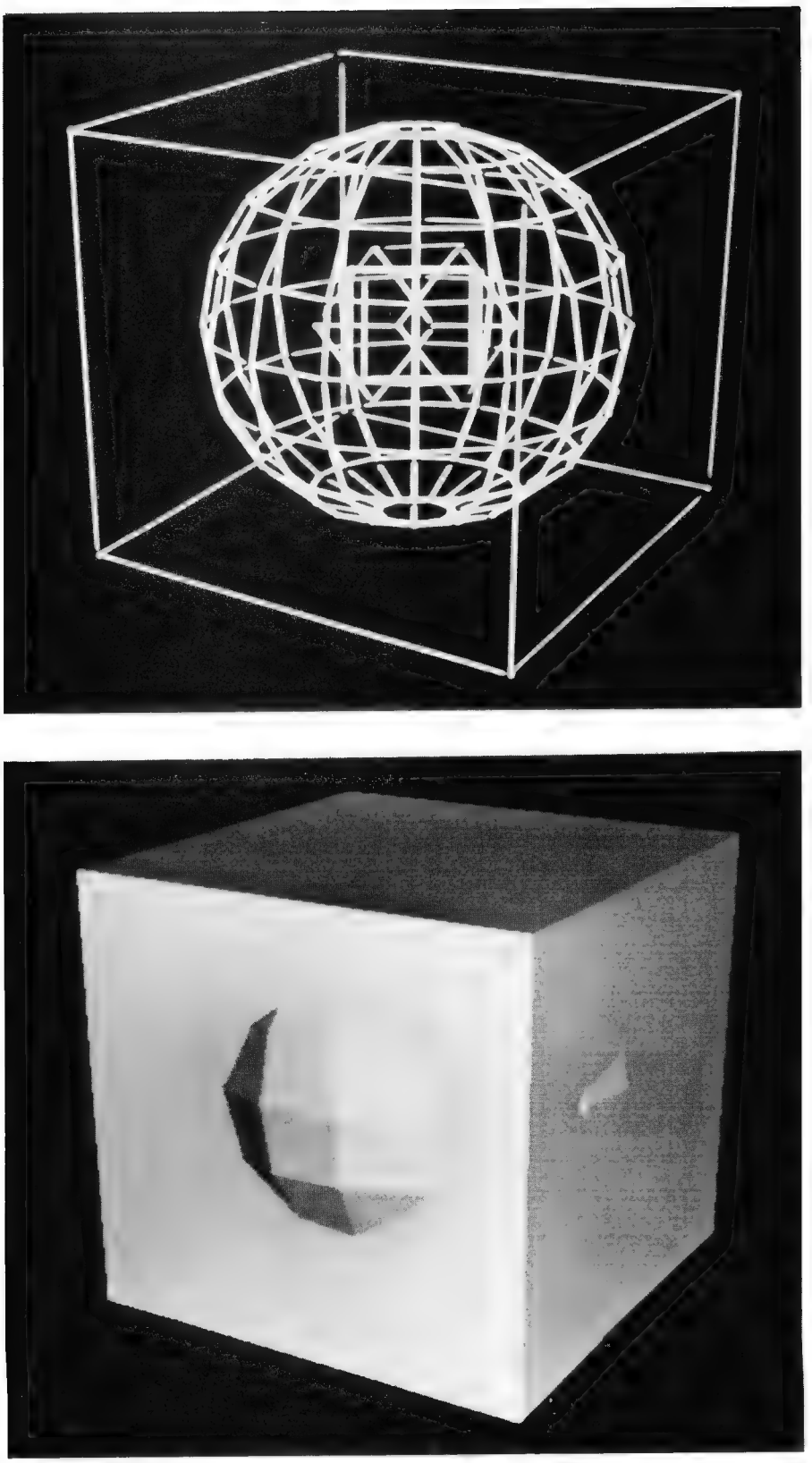


Figure 31
Object 10: Simple2

	VSG1	VSG2	VSG3	VSG4	VSG5	VSG6
1	137607	101892	37919	21730	21224	24291
2	9172	112	38	54	54	56
3	76	44	25	24	23	23
4	7068	6975	6839	9210	-	-
5	26	22	22	15	-	-
6	-	-	-	-	35604	32925

1. Number of memory references required
2. Number of total memory blocks used
3. Maximum number of blocks used at a time
4. Number of multiplications for depth test required (If multiplier-divider used)
5. Number of divisions for intersections required (If multiplier-divider used)
6. Number of addition cycles required for depth comparisons (If multiplier-divider not used)

Figure 32

Statistics of the Penetration Object

for the Six Algorithms

	PENETRATION	E-S	LOW AREA	CUBE1	CUBE2	SHAPE1	SHAPE2	SHEET	SIMPLE1	SIMPLE2
1	49	200	100	150	150	100	100	192	148	148
2	105	480	210	300	300	201	201	308	308	308
3	8802	15278	9154	7158	15410	12424	25590	12500	6172	6664
4	14	0	31	0	0	0	0	0	0	21
5	3844	7052	4193	1238	8221	5630	12023	6935	1652	2383
6	56	246	109	25	121	50	75	250	66	90
7	23	40	31	25	46	24	50	31	18	22
8	24291	47030	24071	7608	80945	34314	79762	26970	9039	14926
9	2.75	3.07	2.62	1.06	5.25	2.76	3.11	2.15	1.46	2.23
10	32925	47174	25641	10374	109190	30227	85941	14176	12326	24983
11	3.38	2.59	3.23	2.89	2.05	2.12	2.66	2.07	3.74	4.94
12	2.96%	3.48%	4.06%	0.00%	4.26%	1.69%	0.52%	1.17%	1.12%	1.07%
13	10	30	10	3	1700	10	1600	25	5	7

Figure 33

Statistics of VSG6 for the Ten Test Objects

1. Number of polygon blocks used to describe the object
2. Number of edge blocks used to describe the object
3. Total number of times that edges cross scan lines
4. Number of implied lines
5. Number of output segments
6. Total number of memory blocks required for storing segment information (300 bits/block)
7. Maximum number of memory blocks ever used at one time for storing segments
8. Total number of references to segment blocks
9. Ratio of line 8 over line 3
10. Total number of add cycles required. (Includes one add cycle per depth test for loading clipping registers)
11. Average number of add cycles per depth test (Includes one add cycle for loading clipping registers)
12. Percent of time when segments are not put on the end of X-sort list for the following scan line
13. Minimum output segment buffer required for real time display

Figure 33 Continued

memory bandwidth was the critical factor, with the polygon segment block being the most accessed array! For the hardware processor, a special purpose memory would be used where 300 bits could be accessed at one time. With semiconductor memories it is becoming economical to do this.

From the first five different changes in the algorithm in Figure 32, one can see a steady decrease in the number of memory references. The final algorithm, however, produced an increase in memory references due to the X-sort technique described in Chapter III-F. In spite of this apparent increase in memory references for VSG6, the overall number of memory references in VSG5 would have been greater if accesses to the bucket X-sort memory had been counted. The design and cost for such a bucket X-sort memory also were compelling factors in deleting it even though accesses to the segment memory increased.

When the program proceeds from one scan line to the next, each segment block needs to be accessed for incrementing the X and Z values. At this same time, another X-sort list is being sorted in preparation for the following scan line. Segments are read from the beginning of the X-sort list for the current scan line and are usually inserted at the end of the X-sort list which is being prepared for the next scan line. Figure 33 (line 12) shows the percentage of times that segments cannot be inserted at the end of the list, and when the previous segment pointers

must be used for finding the correct position in the list for inserting the segment block. The percentage varies between 0 to 4 percent for the ten test objects. Thus, the overhead of tracing back through a list to keep it sorted is extremely low. Also, no large, expensive, or possibly time-consuming special sorting hardware needs to be used.

Visual complexity is much more important in determining the speed of the algorithm than is the total object description. Object 4 and Object 5 are both sets of twenty-five cubes. However, Object 5 requires over ten times the number of memory references as Object 4. A picture visually identical to Object 4, but containing only one cube, was compared with Object 4. Even though Object 4 contained twenty-five cubes, it only had six times the memory references as the single cube object.

One way of measuring the performance of the algorithm is to create a relationship between the object description and the number of memory references to the segment array. Two memory references (a read from memory followed by a write to memory) are always required to increment the X and Z values of a segment when proceeding from one scan line to the next scan line. From the total number of times edges cross scan lines (line 3 of Figure 33), the minimum number of memory references needed can be calculated from Equation 5.

$$M_{min}=(S*N)/E \qquad (5)$$

where M_{min} is the minimum number of memory references that can be expected. S is the number of memory references required to increment a segment (2). N is the total number of times that edges cross scan lines. E is the number of edges contained in a segment (2). Equation 5 reduces to Equation 6.

$$M_{min}=N \tag{6}$$

Equation 7 is the ratio (R) of M_{total} (the total number of memory references actually used) to M_{min}.

$$R=M_{total}/M_{min} \tag{7}$$

Line 9 of Figure 33 lists the different values of R for the ten test objects.

The clipping of segments for depth sorting is very fast. Line 11 of Figure 33 contains the average number of add cycles required by the clipping registers to satisfy the depth comparison test between two polygon segments (see Chapter III-G). One of the add cycles is for loading the clipping registers. Even counting this, the average number is between two to three add cycles per depth test!

The ten test objects were also used by Stephen McCallister [9] for gathering statistics on different versions of Warnock's algorithm. Comparisons are shown for a particular version which divides an area into four sub-areas using a vertex closest to the center of the large area for the common corner of the four sub-areas. If an area is completely covered by a polygon, is void of all

polygons, or has only one visible edge in the area, it is simple enough to be displayed without further reduction.

Statistics for Object 2, E-S (Figure 23), were gathered. A large data structure was used requiring polygon lists, edge lists, and a vertex and planar equation array. Each polygon block consisted of several words, but only accesses to each polygon block (not word) were counted. The same was also true of the remaining data structure. The following information was gathered:

Polygon Block Accesses-----336,156

Edge Block Accesses--------427,688

Vertex Array----------220,910

Planar Equation Array-------21,072

Total Accesses----------1,005,826

The number of accesses to memory was far greater than that required by the new scan line algorithm (47,030). Also, Warnock's algorithm requires that the complete object description be stored in fast memory, and not just those objects pertaining to the current scan line.

D. Output Buffering

Whenever a cathode ray tube (CRT) is being continually refreshed, the rate of moving the beam must remain constant if the displayed intensity is to be a function of the analog input intensity. That is, the X and Y deflection circuits must be changed at a constant rate. The output of the VSG

does not generate segments at a rate inversely proportional to the length of the segments. Therefore, a buffer for temporarily storing segments must be inserted between the VSG and the display.

In Figure 26 (Object 5: Cube2), the Y scan goes from the bottom to the top of the picture. The VSG can quickly determine the visibility of the bottom half of the picture but will require a great amount of time for the top half of the picture. The display, however, must spend the same amount of time on each half of the picture. Because of this, almost the entire bottom half of the picture would need to be buffered. On the other hand in Figure 23 (Object 2: E-S), the VSG runs at a fairly constant rate over the whole picture, and only a small amount of buffering would be required.

For the ten test objects, line 13 of Figure 33 shows the smallest number of segments that must be stored at one time in order to have a display running at thirty frames per second with a constant rate for the X and Y deflection of the CRT beam. The VSG was simulated to reference the polygon segment array every 200 nanoseconds. For objects which have a uniform distribution over the area, only a small buffer size was needed. For Cube2, which has a concentration of visible information in the upper right hand corner, a much larger buffer size was required.

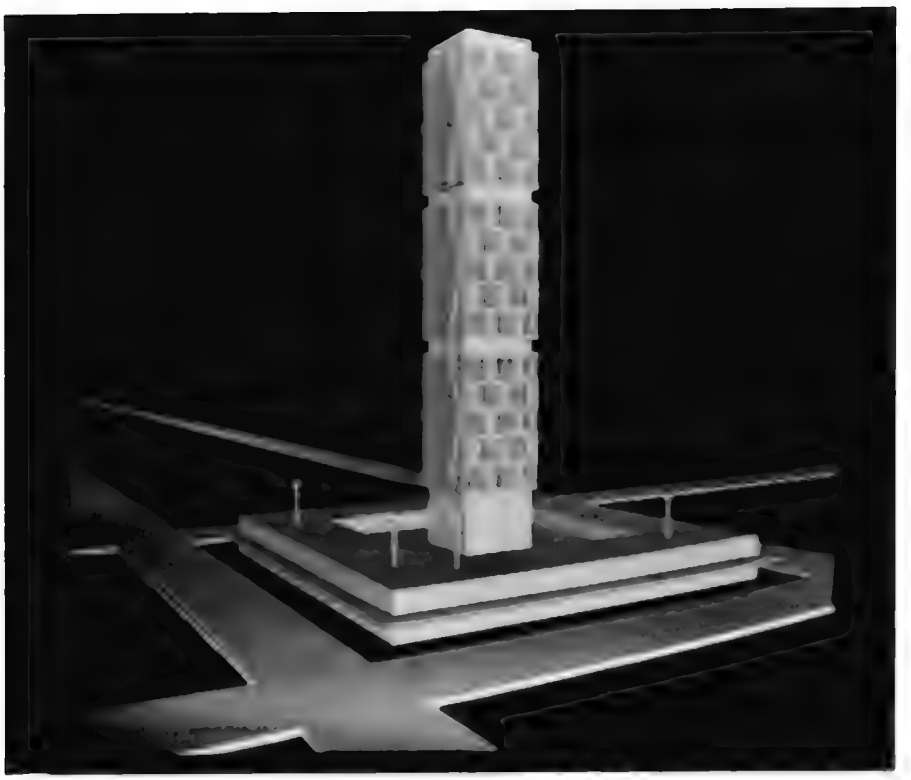


Figure 34
Office Structure

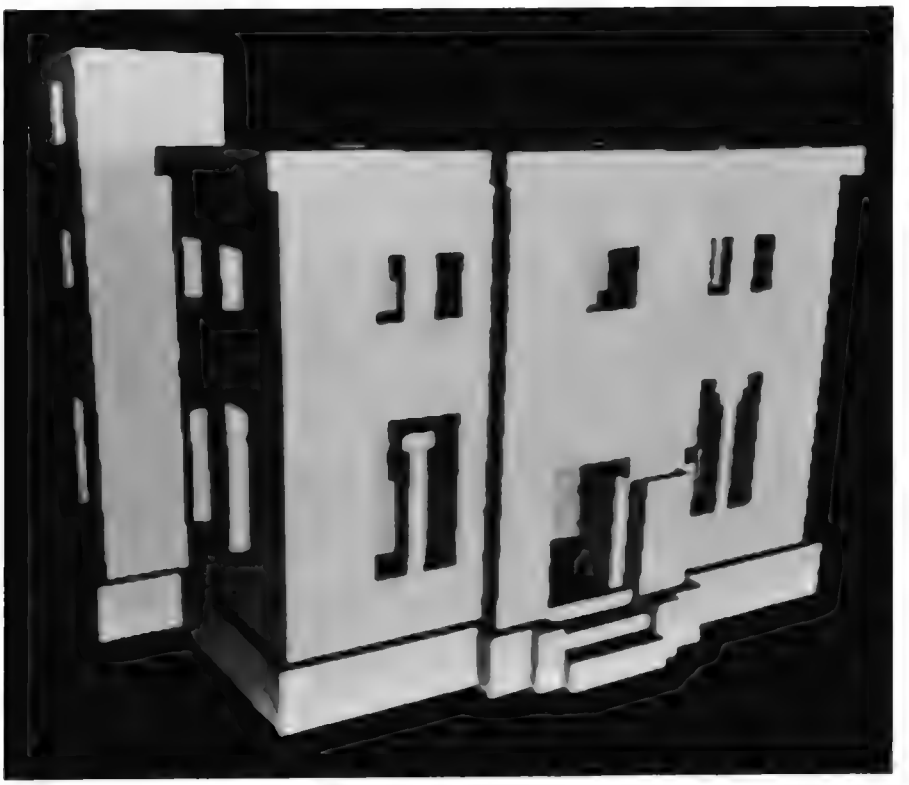


Figure 35
Church

Figure 36
Rear View of Church with Randomly Colored Blocks

Figure 37
Apollo Command and Service Module

Figure 38
Tori

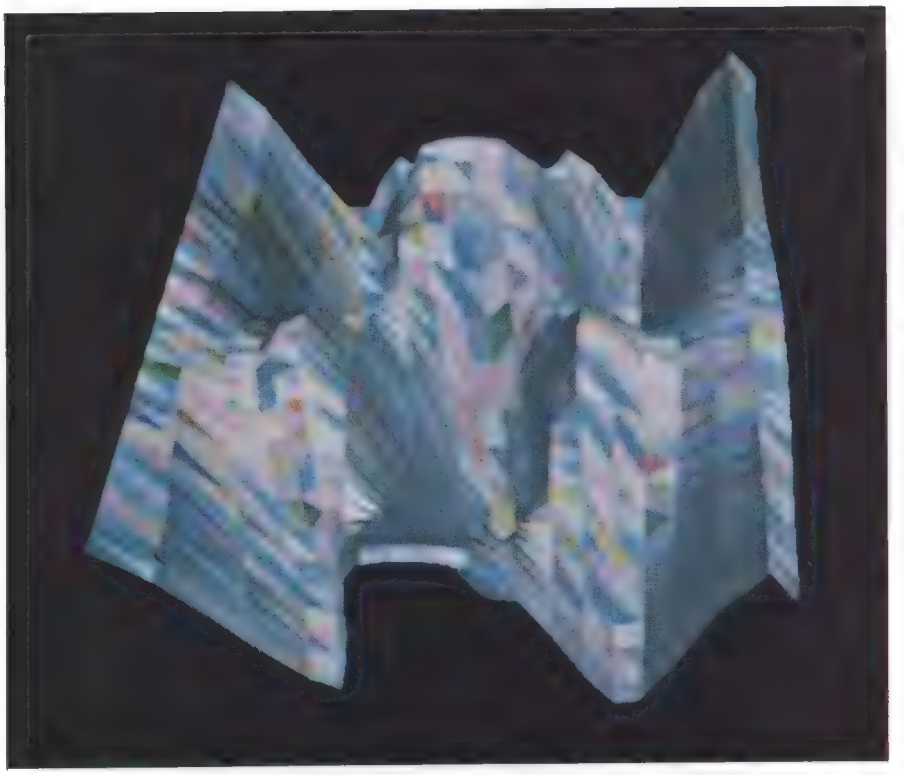


Figure 39
Randomly Colored Surface

CHAPTER VIII

CONCLUSION

The processor described can be built with equipment available today. The segment memory must be in the 200 nanosecond cycle range, and semiconductor memories are available in this range. Also, only a small memory is required since 18 to 50 segment blocks at most are needed at any one time for any of the ten test objects.

The algorithm has been simulated in Fortran IV on a PDP-10 at the Computer Science Department at the University of Utah. Other pictures have been taken to show how coloring and shading adds to the realism of objects. Figures 34-39 show various objects. Total computation time for generating and displaying the pictures is short. Cubel (Object 4) required 30 seconds, and the church of Figure 35 containing 345 blocks (six polygons per block), required only 2.5 minutes. Figure 36 shows the back view of Figure 35 with the blocks randomly colored.

BIBLIOGRAPHY

1. Roberts, L. G. "Machine Perception of Three-Dimensional Solids," Technical Report No. 315, Lincoln Laboratory, M.I.T., Cambridge, Mass., 22 May 1963.

2. Loutrel, P. P. "A Solution to the Hidden-Line Problem for Computer-Drawn Polyhedra," IEEE Transactions on Computers, C-19 [3], 205 March 1970.

3. Appel, A. "The Notion of Quantitative Invisibility and the Machine Rendering of Solids," ACM Conference Proc. 387 (1967).

4. Wylie, C., Romney, G., Evans, D. C., Erdahl, A. "Half-tone Perspective Drawings by Computer," AFIPS Proc. FJCC 31, 49 November 1967.

5. Romney, G. "Computer Assisted Assembly and Rendering of Solids," Computer Science, University of Utah, Salt Lake City, Utah, August 1969.

6. Warnock, J. "A Hidden Surface Algorithm for Computer Generated Halftone Pictures," Technical Report 4-15, Computer Science, University of Utah, Salt Lake City, Utah, June 1969.

7. Bouknight, W. J. "An Improved Procedure for Generation of Half-tone Computer Graphics Presentations," Report R-432, Coordinated Science Laboratory, University of Illinois, Urbana, Illinois, September 1969.

8. Sproull, R., Sutherland, I. E. "A Clipping Divider," AFIPS Proc. FJCC 33, 765 (1968).

9. McCallister, S., Sutherland, I. E. "Final Report on the Area Warnock Hidden Line Algorithm," Evans and Sutherland Computer Corporation, Salt Lake City, Utah, Internal Document, 12 February 1970.

APPENDIX I

LISTING OF PROGRAM

The hidden line program is called as a subroutine from a main program. VSG6 contains counters interspersed throughout the program for gathering statistics like those in Appendix II. VSG6 is written in FORTRAN IV.

Several subroutines are called by the program:

LDRPT(I,J) < loads the right half of J (sign extended) into I.

LDLPT(I,J) < loads the left half of J (sign extended) into I.

STRPT(I,J) < stores the right half of I into the right half of J. The left half of J remains undisturbed.

STLPT(I,J) < stores the right half of I into the left half of J. The right half of J remains undisturbed.

SHOW < displays the segments stored in the VISSEG array.

LSTSET(N) < initializes a free list structure with blocks of N words each.

GETBLK(I) < gets a block from the free list. I is the index of that block and is set by the subroutine.

RETBLK(I) < returns a block to the free list. I is the index of the block to be returned.

```
      SUBROUTINE HIDDEN(PIX,STAT)
      COMMON/FREE/EDGEST,DUM,POLYST
      COMMON/FREE1/Q1(4),FRAMEX,FRAMEY
      COMMON/FREE2/X(1000),Y(1000),Z(1000),CX(700),CY(700),
     1CZ(700),CD(700)
      IMPLICIT INTEGER (A-Z)
      REAL X,Y,Z,CX,CY,CZ,CD
      COMMON/SCOPE/VISSEG(512),BUCKY(512)
      DIMENSION EDGE(1),SEG(1),POLY(1)
      EQUIVALENCE (EDGEST,EDGE,SEG,POLY)
      DIMENSION ZS(10),SAM(3,2)
      DIMENSION PQ(16),Q(10,26),ADDS(20)
      PQL=16
      QL=26
C
C     PQ(1)=NUMBER OF TOTAL BLOCKS REQUIRED FOR
C          HIDDEN LINE WORK.
C     *PQ(2)=MAXIMUM NUMBER OF TOTAL BLOCKS EVER USED AT ONE TIME.
C
C     PQ(3)=CURRENT NUMBER OF TOTAL BLOCKS AT A GIVEN TIME.
C          (USED FOR CALCULATING PQ(2).)
C     PQ(4)=TOTAL NUMBER OF EDGE BLOCKS IN FRAME.
C
C     PQ(5)=NUMBER OF EDGE BLOCKS WITH AT LEAST ONE OF THE
C           CONNECTED POLYGONS DRAWN CLOCKWISE.
C     PQ(6)=NUMBER OF THOSE EDGE BLOCKS OF PQ(5) WHOSE Y VALUE
C           OF THE BEGIN PT IS NOT THE SAME AS THE END PT Y VALUE.
C     PQ(7)=TOTAL NUMBER OF POLYGON BLOCKS IN THE FRAME.
C
C     PQ(8)=NUMBER OF POLYGON BLOCKS DRAWN CLOCKWISE.
C
C     PQ(9)=POINT DENSITY.
C
C     PQ(10)=NUMBER OF INVOLVED SCAN LINES.
C
C     PQ(11)=MEMORY REFERENCES FOR SEGMENT CREATOR.
C
C     PQ(12)=NANOSECONDS PER MEMORY REFERENCE FOR SEGMENT CREATOR.
C
C     PQ(13)=MEMORY REFERENCES FOR DEPTH CALCULATOR.
C
C     PQ(14)=NANOSECONDS PER MEMORY REFERENCE FOR DEPTH CALCULATOR.
C
C     PQ(15)=MEMORY REF. TOTAL PQ(11),PQ(13)
C
C     PQ(16)=NANOSECONDS FOR PQ(15).
C
C     ADDS(I)=NUMBER OF TIMES THE DEPTH TEST WAS SATISFIED IN.
```

```
C       Q COUNTERS
C       Q(1,X)=TOTAL PER FRAME
C       Q(2,X)=MAXIMUM REQUIRED OF A SCAN LINE
C       Q(3,X)=AVERAGE OF TOTAL SCAN LINES. ALSO SCRATCH FOR Q(2,X)
 C      Q(4,X)=REQUIRED FOR PRE-FRAME PROCESSING
C       Q(5,X)=MAXIMUM REQUIRED FOR SCAN PREPARATION PROCESSING
C       Q(6,X)=NANOSECONDS REQUIRED. ALSO SCRATCH FOR Q(5,X)
C       Q(7,X)=MAXIMUM REQUIRED FOR SCAN DEPTH PROCESSING
C       Q(8,X)=AVERAGE OF ACTIVE SCAN LINES. ALSO SCRATCH FOR Q(7,X)
 C      Q(9,X)=TOTAL FOR SCAN PREPARATION PROCESSING
C       Q(10,X)=TOTAL FOR SCAN DEPTH PROCESSING
C
C
C       Q(X,1)=NUMBER OF SLOPE CALCULATIONS.
C       Q(X,2)=NUMBER OF INTERCEPT CALCULATIONS.
C       Q(X,3)=NUMBER OF SAMPLE POINTS STORED FOR NEXT SCAN LINE.
C       Q(X,4)=SUBDIVISIONS (NOT FROM INTERSECTING CASE).
C       Q(X,5)=
C       Q(X,6)=DEPTH SAMPLES REQUIRED.
C       Q(X,7)=
C       Q(X,8)=SAMPLE POINTS DELETED.
C       Q(X,9)=
C       Q(X,10)=OUTPUT SEGMENTS.
C       Q(X,11)=INTERCEPT CALCULATIONS.
C       Q(X,12)=INTERCEPT SUBDIVISIONS.
C       Q(X,13)=OVERHEAD PIPELINE TIME.
C       Q(X,14)=TIME WAITING FOR CLIPPER.
C       Q(X,15)=READS FROM POLY.
C       Q(X,16)=WRITES TO POLY.
C       Q(X,17)=READS FROM EDGE.
C       Q(X,18)=WRITES TO EDGE.
C       Q(X,19)=READS FROM SEG
C       Q(X,20)=WRITES TO SEG
C       Q(X,21)=
C       Q(X,22)=
C       Q(X,23)=READS FROM FREE LIST(GETBLK)
C       Q(X,24)=WRITES TO FREE LIST(RETBLK)
C       Q(X,25)=READS FROM BUCKY
C       Q(X,26)=USED FOR SHADER
```

```
C       INITIALIZATION.
C
        DO 8 I=1,QL
        DO 8 J=1,10
8       Q(J,I)=0
        DO 9 I=1,PQL
9       PQ(I)=0
        DO 12 I=1,20
12      ADDS(I)=0
        DO 10 I=1,FRAMEY
10      BUCKY(I)=0
        CALL LSTSET(14)
        DEPTH=.TRUE.
        SAM2S=0
        SAM2X=FRAMEX
        SEGS2=0
        SEGL2=0
        POLYCH=0
        IMPLST=0
C       GO THROUGH ALL POLYGONS AND NOTE WHICH WAY EACH POLYGON
C       IS DRAWN (CLOCKWISE OR COUNTER CLOCKWISE) BY CHECKING
C       CZ OF PLANAR EQUATIONS AND MARK THE POLYGON BLOCK.
        POLYPT=POLYST
90      IF(POLYPT.EQ.0)GO TO 99
        POLY(POLYPT+1)=-1
        CALL LDRPT(INDEX,POLY(POLYPT+2))
        Q(1,15)=Q(1,15)+1
        PQ(7)=PQ(7)+1
        Q(1,16)=Q(1,16)+1
        IF(CZ(INDEX).LE.0)GO TO 95
        POLY(POLYPT+1)=0
        POLY(POLYPT+3)=0
        PQ(8)=PQ(8)+1
95      CALL LDRPT(POLYPT,POLY(POLYPT))
        GO TO 90
```

```
C       INITIALIZATION CONTINUED.
C       TAKE EACH EDGE AND PUT IN THE BUCKY GIVEN BY ITS
C       SMALLEST Y VALUE.  THIS IS THE Y-SORT OF EDGES.
99      EDGEPT=EDGEST
100     IF(EDGEPT.EQ.0)GO TO 200
        PQ(4)=PQ(4)+1
C
C       ENTER EACH EDGE IN BUCKY IF AT LEAST ONE OF THE
C       TWO POLYGONS IS DRAWN CLOCKWISE.
C
        CALL LDLPT(POLYL,EDGE(EDGEPT+2))
        CALL LDRPT(POLYR,EDGE(EDGEPT+2))
        Q(1,17)=Q(1,17)+1
        IF(POLYR.EQ.POLYL)GO TO 110
        IF(POLYL.EQ.0)GO TO 103
        IF(POLY(POLYL+1).EQ.0)GO TO 104
103     IF(POLY(POLYR+1).LT.0)GO TO 110
        Q(1,15)=Q(1,15)+1
104     CALL LDLPT(INDEX,EDGE(EDGEPT+1))
        YBEG=Y(INDEX)
        CALL LDRPT(INDEX,EDGE(EDGEPT+1))
        YEND=Y(INDEX)
        PQ(5)=PQ(5)+1
        IF(YBEG.EQ.YEND)GO TO 110
        PQ(6)=PQ(6)+1
        IF(YBEG.LT.YEND)GO TO 105
        I=YEND
        YEND=YBEG
        YBEG=I
105     YBEG=YBEG+1
        IF(YBEG.LE.0)GO TO 115
        IF(YEND.GE.FRAMEY)GO TO 115
        I=BUCKY(YBEG)
        BUCKY(YBEG)=EDGEPT
        Q(1,18)=Q(1,18)+1
        CALL STLPT(I,EDGE(EDGEPT))
110     CALL LDRPT(EDGEPT,EDGE(EDGEPT))
        GO TO 100
 15     TYPE 116
        RETURN
 16     FORMAT(' ERROR...OBJECT NOT IN BOUNDS OF FRAME!')
```

```
C       SCAN LINE COMPUTATION.
200     CONTINUE
        DO 201 I=1,QL
201     Q(4,I)=Q(1,I)
        IY=0
204     IY=IY+1
        DO 202 I=1,QL
        Q(6,I)=Q(1,I)
        Q(9,I)=Q(9,I)-Q(1,I)
202     Q(3,I)=Q(1,I)
C       INITIALIZE ALL POINTERS.
        SEGXST=SEGS2
        SEGLST=SEGL2
        SEGS2=0
        SEGL2=0
        SAM1S=SAM2S
        SAM1L=SAM2L
        SAM2S=0
        SAM1X=SAM2X
        SAM2X=FRAMEX
        IF(IY.GT.FRAMEY)GO TO 230
        SEGCNT=0
C       SCAN PREPARATION PROCESSING.
C       GET EDGES FROM BUCKY WHICH ENTER ON THIS SCAN LINE
C       AND BUILD THE SEGMENT LIST (SEG).
        IF(BUCKY(IY).EQ.0)GO TO 230
        Q(1,25)=Q(1,25)+1
        EDGEPT=BUCKY(IY)
210     IF(EDGEPT.EQ.0)GO TO 230
        Q(1,17)=Q(1,17)+1
        CALL LDLPT(BEG,EDGE(EDGEPT+1))
        CALL LDRPT(END,EDGE(EDGEPT+1))
        YEND=Y(END)
        YBEG=Y(BEG)
        DELY=YBEG-YEND
        IF(DELY.EQ.0)GO TO 229
        IF(DELY.LT.0)GO TO 211
        I=BEG
        BEG=END
        END=I
        DELY=-DELY
211     IX=X(BEG)*262144.0
        Q(1,1)=Q(1,1)+1
        XSLOPE=((X(END)-X(BEG))/(Y(END)-Y(BEG)))*262144.0
        DEL=(IY-Y(BEG))*262144.0
        IX=IX+ZMUL(XSLOPE,DEL)
        CALL LDLPT(IXE,IX)
        IF(IXE.LE.0.OR.IXE.GT.FRAMEX)GO TO 115
        II=-1
        CALL LDRPT(POLYPT,EDGE(EDGEPT+2))
C       TWO POLYGONS PER EDGE ARE ALLOWED.
212     IF(POLYPT.EQ.0)GO TO 228
        Q(1,15)=Q(1,15)+1
        IF(POLY(POLYPT+1).EQ.-1)GO TO 228
        Q(1,16)=Q(1,16)+1
        IF(POLY(POLYPT+1).LT.0)GO TO 213
        POLY(POLYPT+1)=POLYCH
        POLYCH=POLYPT
        CALL STLPT(-1,POLY(POLYPT+1))
213     SEGPT=POLY(POLYPT+3)
        PREV=0
        YEND2P=-1
```

```
214     IF(SEGPT.EQ.0)GO TO 220
        Q(1,19)=Q(1,19)+1
        CALL LDRPT(YEND2,SEG(SEGPT+2))
        CALL LDLPT(YEND1,SEG(SEGPT+2))
        IF(YEND1.GE.0)GO TO 217
        TE1=IX-SEG(SEGPT+3)-SEG(SEGPT+4)
        IF(TE1.EQ.0)TE1=XSLOPE-SEG(SEGPT+4)
        IF(TE1.LT.0)GO TO 220
        IF(YEND2.GE.0)GO TO 218
        TE2=IX-SEG(SEGPT+5)-SEG(SEGPT+6)
        IF(TE2.EQ.0)TE2=XSLOPE-SEG(SEGPT+6)
        IF(TE2.LT.0)GO TO 223
        GO TO 218
217     IF(YEND2.GE.0)GO TO 218
        TE2=IX-SEG(SEGPT+5)-SEG(SEGPT+6)
        IF(TE2.EQ.0)TE2=XSLOPE-SEG(SEGPT+6)
        IF(TE2.GE.0)GO TO 218
        MODE=0
        PREV=SEGPT
        GO TO 227
218     YEND2P=YEND2
        PREV=SEGPT
        CALL LDRPT(SEGPT,SEG(SEGPT+1))
        GO TO 214
220     MODE=2
        IF(YEND2P.GE.0)GO TO 227
        FROM=0
        GO TO 226
```

```
223     FROM=-1
        PREV=SEGPT
        CALL LDRPT(SEGPT,SEG(SEGPT+1))
        GO TO 226
224     SEG(I+5)=SEG(PREV+5)
        SEG(I+6)=SEG(PREV+6)
        SEG(I+9)=SEG(PREV+9)
        SEG(I+10)=SEG(PREV+10)
        Q(1,20)=Q(1,20)+1
        CALL STRPT(YEND2,SEG(I+2))
        MODE=2
        GO TO 227
226     CALL GETBLK(I)
        Q(1,23)=Q(1,23)+1
        PQ(1)=PQ(1)+1
        PQ(3)=PQ(3)+1
        IF(PQ(3).GT.PQ(2))PQ(2)=PQ(3)
        CALL STRPT(SEGPT,SEG(I+1))
        IF(PREV.NE.0)CALL STRPT(I,SEG(PREV+1))
        IF(PREV.NE.0)Q(1,20)=Q(1,20)+1
        IF(PREV.EQ.0)POLY(POLYPT+3)=I
        SEG(I)=-1
        SEG(I+2)=0
        SEG(I+11)=0
        CALL STLPT(POLYPT,SEG(I+1))
        IF(FROM.NE.0)GO TO 224
        PREV=I
227     SEG(PREV+3+MODE)=IX-XSLOPE
        SEG(PREV+4+MODE)=XSLOPE
        Q(1,20)=Q(1,20)+1
        IF(MODE.EQ.0)CALL STLPT(DELY,SEG(PREV+2))
        IF(MODE.NE.0)CALL STRPT(DELY,SEG(PREV+2))
        SEG(PREV+8+MODE)=((Z(END)-Z(BEG))/(Y(END)-Y(BEG)))*262144.0
        SEG(PREV+7+MODE)=Z(BEG)*262144.0
        SEG(PREV+7+MODE)=SEG(PREV+7+MODE)+ZMUL(SEG(PREV+8+MODE),DEL)
        SEG(PREV+7+MODE)=SEG(PREV+7+MODE)-SEG(PREV+8+MODE)
228     CALL LDLPT(POLYPT,EDGE(EDGEPT+2))
        II=II+1
        IF(II.EQ.0)GO TO 212
229     CALL LDLPT(EDGEPT,EDGE(EDGEPT))
        GO TO 210
```

```
C       SEGMENT PACKER AND SEGMENT ELIMINATOR.
230     IF (POLYCH.EQ.0) GO TO 242
        Q(1,15)=Q(1,15)+1
        Q(1,16)=Q(1,16)+1
        CALL STLPT(0,POLY(POLYCH+1))
        NEXT=POLY(POLYCH+3)
        SEGPT=0
231     IF (NEXT.EQ.0) GO TO 240
        PREV=SEGPT
        SEGPT=NEXT
        Q(1,19)=Q(1,19)+1
        CALL LDRPT(NEXT,SEG(SEGPT+1))
        IF (SEG(SEGPT+2).NE.0) GO TO 233
        IF(PREV.NE.0)Q(1,20)=Q(1,20)+1
        IF(PREV.NE.0)CALL STRPT(NEXT,SEG(PREV+1))
        IF(PREV.EQ.0)POLY(POLYCH+3)=NEXT
        Q(1,24)=Q(1,24)+1
        PQ(3)=PQ(3)-1
        CALL RETBLK(SEGPT)
        SEGPT=PREV
        GO TO 231
233     NEXT1=NEXT
        CALL LDRPT(YEND2,SEG(SEGPT+2))
        IF (YEND2.GE.0) GO TO 237
        CALL LDLPT(YEND1,SEG(SEGPT+2))
        IF(YEND1.LT.0)GO TO 2395
        SEG(SEGPT+3)=SEG(SEGPT+5)
        SEG(SEGPT+4)=SEG(SEGPT+6)
        SEG(SEGPT+7)=SEG(SEGPT+9)
        SEG(SEGPT+8)=SEG(SEGPT+10)
        CALL STLPT(YEND2,SEG(SEGPT+2))
```

```
237     IF(NEXT1.EQ.0)GO TO 241
        CALL LDLPT(YEND1,SEG(NEXT1+2))
        Q(1,19)=Q(1,19)+1
        IF (YEND1.GE.0) GO TO 238
        Q(1,20)=Q(1,20)+2
        CALL STRPT(YEND1,SEG(SEGPT+2))
        CALL STLPT(0,SEG(NEXT1+2))
        SEG(SEGPT+5)=SEG(NEXT1+3)
        SEG(SEGPT+6)=SEG(NEXT1+4)
        SEG(SEGPT+9)=SEG(NEXT1+7)
        SEG(SEGPT+10)=SEG(NEXT1+8)
        CALL LDLPT(S1,SEG(NEXT1))
        CALL LDRPT(S2,SEG(NEXT1))
        IF(S1.NE.0)CALL STRPT(S2,SEG(S1))
        IF(S1.EQ.0)SEGXST=S2
        IF(NEXT1.NE.SEGLST)CALL STLPT(S1,SEG(S2))
        IF(NEXT1.EQ.SEGLST)SEGLST=S1
        Q(1,20)=Q(1,20)+1
        SEG(NEXT1)=-1
        GO TO 2395
238     CALL LDRPT(YEND2,SEG(NEXT1+2))
        IF (YEND2.GE.0) GO TO 239
        Q(1,20)=Q(1,20)+2
        CALL STRPT(YEND2,SEG(SEGPT+2))
        SEG(NEXT1+2)=0
        SEG(SEGPT+5)=SEG(NEXT1+5)
        SEG(SEGPT+6)=SEG(NEXT1+6)
        SEG(SEGPT+9)=SEG(NEXT1+9)
        SEG(SEGPT+10)=SEG(NEXT1+10)
        GO TO 2395
239     CALL LDRPT(NEXT1,SEG(NEXT1+1))
        GO TO 237
2395    IF(SEG(SEGPT).NE.-1)GO TO 231
        CALL LDLPT(IXE,SEG(SEGPT+3)+SEG(SEGPT+4))
        S1=PREV
2396    IF(S1.NE.0)CALL LDRPT(S2,SEG(S1))
        IF(S1.EQ.0)S2=SEGXST
        IF(S1.EQ.SEGLST)S2=0
        Q(1,19)=Q(1,19)+1
        IF(S2.EQ.0)GO TO 2397
        CALL LDLPT(IX,SEG(S2+3)+SEG(S2+4))
        IF(IX.GE.IXE)GO TO 2397
        S1=S2
        GO TO 2396
2397    IF(S2.NE.0)SEG(SEGPT)=S2
        Q(1,20)=Q(1,20)+1
        CALL STLPT(S1,SEG(SEGPT))
        IF(S1.NE.0)CALL STRPT(SEGPT,SEG(S1))
        IF(S1.EQ.0)SEGXST=SEGPT
        IF(S2.NE.0)CALL STLPT(SEGPT,SEG(S2))
        IF(S2.EQ.0)SEGLST=SEGPT
        GO TO 231
240     POLYCH=POLY(POLYCH+1)
        GO TO 230
241     PAUSE 'UNCLOSED POLYGON'
        SEG(SEGPT+5)=SEG(SEGPT+3)
        SEG(SEGPT+6)=SEG(SEGPT+4)
        CALL STRPT(0,SEG(SEGPT+2))
        GO TO 2395
```

```
C       DEPTH SORTER.
242     CONTINUE
        DO 276 I=1,QL
        Q(6,I)=Q(1,I)-Q(6,I)
        IF(Q(5,I).LT.Q(6,I))Q(5,I)=Q(6,I)
        Q(9,I)=Q(9,I)+Q(1,I)
        Q(10,I)=Q(10,I)-Q(1,I)
276     Q(8,I)=Q(1,I)
        IF(IY.GT.FRAMEY)GO TO 498
        SAM(1,2)=1
        SAM(2,2)=0
        CALL LDLPT(SEGPT,IMPLST)
278     IF(SEGPT.EQ.0)GO TO 279
        NEXT=SEG(SEGPT)
        CALL RETBLK(SEGPT)
        PQ(3)=PQ(3)-1
        Q(1,24)=Q(1,24)+1
        SEGPT=NEXT
        GO TO 278
279     IMPLST=IMPLST*262144
        SEGACT=0
C               SAMPLE SPAN GENERATOR.
281     SAM(1,1)=SAM(1,2)+1
        SAM(2,1)=SAM(2,2)
        SAM(3,1)=SAM(3,2)
        SAM(2,2)=0
        IF(SAM1X.GE.SAM(1,1))GO TO 282
        SAM1X=FRAMEX
        IF(SAM1S.EQ.SAM1L)GO TO 282
        CALL LDLPT(SAM1X,SEG(SAM1S))
        CALL LDRPT(SAM1S,SEG(SAM1S))
282     SAM(1,2)=SAM1X
299     ZS(1)=0
        FROM=0
        SEGPT=SEGACT
        SEGOUT=0
        PREV=0
```

```
C       CHECK SEGMENTS FROM THE CURRENT ACTIVE LIST.
301     IF (SEGPT.EQ.0) GO TO 304
        NUMREF=-Q(1,19)-Q(1,20)-Q(1,13)
        Q(1,19)=Q(1,19)+1
        NEXT=SEG(SEGPT+11)
        XLEFT=SEG(SEGPT+3)
        XRIGHT=SEG(SEGPT+5)
        ZLEFT=SEG(SEGPT+7)
        ZRIGHT=SEG(SEGPT+9)
        CALL LDLPT(IXE,XLEFT)
        CALL LDLPT(IXX,XRIGHT)
        IF(IXX.LE.SAM(1,2))GO TO 303
        PREV=SEGPT
        IF(IXE.GE.SAM(1,2))GO TO 335
        GO TO 315
303     CONTINUE
        Q(1,20)=Q(1,20)+1
        IF(PREV.NE.0)SEG(PREV+11)=NEXT
        IF(PREV.EQ.0)SEGACT=NEXT
        IF (IXX.LT.SAM(1,1)) GO TO 335
        Q(1,20)=Q(1,20)+1
        SEG(SEGPT+11)=SEGOUT
        IF (SEGOUT.EQ.0) SEGLO=SEGPT
        SEGOUT=SEGPT
        GO TO 315
C       CHECK NEW SEGMENTS FROM THE X-SORT LIST. ALSO
C       INCREMENT THE X,Y,Z VALUES AND INSERT THE SEGMENT BLOCK
C       IN THE X-SORT LIST FOR THE NEXT SCAN LINE.
304     SEGPT=SEGXST
        IF(SEGPT.EQ.0)GO TO 350
        NUMREF=-Q(1,19)-Q(1,20)-Q(1,13)
        Q(1,19)=Q(1,19)+1
        CALL LDLPT(IXE,SEG(SEGPT+3)+SEG(SEGPT+4))
        IF(IXE.GE.SAM(1,2))GO TO 350
        FROM=-1
        CALL LDRPT(SEGXST,SEG(SEGPT))
        IF(SEGPT.EQ.SEGLST)SEGXST=0
        Q(1,2)=Q(1,2)+1
        SEG(SEGPT+3)=SEG(SEGPT+3)+SEG(SEGPT+4)
        SEG(SEGPT+5)=SEG(SEGPT+5)+SEG(SEGPT+6)
        SEG(SEGPT+7)=SEG(SEGPT+7)+SEG(SEGPT+8)
        SEG(SEGPT+9)=SEG(SEGPT+9)+SEG(SEGPT+10)
        XLEFT=SEG(SEGPT+3)
        XRIGHT=SEG(SEGPT+5)
        ZLEFT=SEG(SEGPT+7)
        ZRIGHT=SEG(SEGPT+9)
        CALL LDLPT(YEND1,SEG(SEGPT+2))
        CALL LDRPT(YEND2,SEG(SEGPT+2))
        YEND1=YEND1+1
        YEND2=YEND2+1
        CALL STLPT(YEND1,SEG(SEGPT+2))
        CALL STRPT(YEND2,SEG(SEGPT+2))
```

```
      IF (SEG(SEGPT+11).GE.0) GO TO 309
      IF (YEND2.GE.0) GO TO 308
      IF (IXE+1.NE.SAM(1,1)) GO TO 308
      CALL LDLPT(IX,SEG(SEGPT+3)+SEG(SEGPT+4))
      IF (IX.LE.0.OR.IX.GT.FRAMEX) GO TO 308
      SAM(3,1)=SEGPT+12
      SAM(2,1)=IX
      FM=-1
      GO TO 3091
308   CALL RETBLK(SEGPT)
      PQ(3)=PQ(3)-1
      Q(1,24)=Q(1,24)+1
      GO TO 335
309   MODE=0
      SEG(SEGPT)=-1
      IF (YEND1.GE.0) GO TO 310
      MODE=-1
      CALL LDLPT(IX,SEG(SEGPT+3)+SEG(SEGPT+4))
      IF (IX.LE.0.OR.IX.GT.FRAMEX) GO TO 115
      FM=0
3091  S2=0
      S1=SEGL2
3092  IF(S1.EQ.0)GO TO 3094
      CALL LDLPT(IX1,SEG(S1+3)+SEG(S1+4))
      IF(IX.GE.IX1)GO TO 3094
      S2=S1
      CALL LDLPT(S1,SEG(S1))
      Q(1,19)=Q(1,19)+1
      GO TO 3092
3094  IF(S2.NE.0)SEG(SEGPT)=S2
      Q(1,20)=Q(1,20)+1
      CALL STLPT(S1,SEG(SEGPT))
      IF(S2.NE.0)CALL STLPT(SEGPT,SEG(S2))
      IF(S2.EQ.0)SEGL2=SEGPT
      IF(S1.NE.0)CALL STRPT(SEGPT,SEG(S1))
      IF(S1.EQ.0)SEGS2=SEGPT
      IF(S2.NE.0)Q(1,20)=Q(1,20)+1
      IF(FM)335,310,364
310   MODE=-MODE
      IF (YEND2.GE.0) GO TO 311
      MODE=-MODE
      CALL LDLPT(IX,SEG(SEGPT+5)+SEG(SEGPT+6))
      IF (IX.LE.0.OR.IX.GT.FRAMEX) GO TO 115
311   IF (MODE.LT.0) GO TO 312
C     IF EITHER OF THE EDGES OF THE SEGMENT EXIT ON THIS
C     SCAN LINE (MODE), PUT THE CORRESPONDING POLYGON IN
C     THE POLYGON CHANGING LIST.
      CALL LDLPT(POLYPT,SEG(SEGPT+1))
      Q(1,15)=Q(1,15)+1
      IF (POLY(POLYPT+1).LT.0) GO TO 312
      Q(1,16)=Q(1,16)+1
      POLY(POLYPT+1)=POLYCH
      POLYCH=POLYPT
      CALL STLPT(-1,POLY(POLYPT+1))
312   CALL LDLPT(IXX,XRIGHT)
      IF (IXE.GE.IXX) GO TO 335
      IF (IXX.GT.SAM(1,2)) GO TO 314
      SEG(SEGPT+11)=SEGOUT
      IF (SEGOUT.EQ.0) SEGLO=SEGPT
      SEGOUT=SEGPT
      GO TO 315
314   SEG(SEGPT+11)=SEGACT
      SEGACT=SEGPT
```

```
315     CONTINUE
        Q(1,6)=Q(1,6)+1
        IXLEFT=IXE
        IF(IXE.LT.SAM(1,1))IXLEFT=IXX
        CALL LDLPT(YEND1,SEG(SEGPT+2))
        IF(YEND1.GE.0)GO TO 316
        IF(IXE+1.NE.SAM(1,1))GO TO 316
        SAM(3,1)=SEGPT+12
        CALL LDLPT(SAM(2,1),SEG(SEGPT+3)+SEG(SEGPT+4))
316     CALL LDRPT(YEND2,SEG(SEGPT+2))
        IF(YEND2.GE.0)GO TO 317
        IF(IXX.NE.SAM(1,2))GO TO 317
        SAM(3,2)=SEGPT+13
        CALL LDLPT(SAM(2,2),SEG(SEGPT+5)+SEG(SEGPT+6))
C       SIMULATION OF LOADING AND RUNNING THE CLIPPER ARTHIMETIC
C       UNIT FOR DEPTH COMPARISONS.
C       ADDITION TIME ONE.
317     XLTEST=XLEFT.AND.(.NOT.262143)
        XRTEST=XRIGHT.AND.(.NOT.262143)
        NUMADD=1
        IF(FROM.NE.0)NUMADD=NUMADD+1
        DELNEW=(XRTEST-XLTEST)/262144
        IF(.NOT.DEPTH)DELNEW=DELNEW*1024
        IF((XLTEST-SAM(1,1)*262144).LT.0)XLTEST=SAM(1,1)*262144
        IF((XRTEST-SAM(1,2)*262144).GE.0)XRTEST=SAM(1,2)*262144
        ADJNEW=.FALSE.
        IF(ZLEFT.LT.ZRIGHT)ADJNEW=.TRUE.
        IF(ZS(1).EQ.0)GO TO 331
        ABLLE=.FALSE.
        ABLGE=.FALSE.
        ABRLE=.FALSE.
        ABRGE=.FALSE.
        IF(XLTEST.LE.ZS(6))ABLLE=.TRUE.
        IF(XLTEST.GE.ZS(6))ABLGE=.TRUE.
        IF(XRTEST.LE.ZS(7))ABRLE=.TRUE.
        IF(XRTEST.GE.ZS(7))ABRGE=.TRUE.
        IF(((.NOT.ABLGE).AND.(.NOT.ABRGE)).OR.((.NOT.ABLLE).AND.
        1(.NOT.ABRLE)))GO TO 329
        XLCLIP=XLTEST
        IF(ABLLE)XLCLIP=ZS(6)
        XRCLIP=XRTEST
        IF(ABRGE)XRCLIP=ZS(7)
        DEL=DELNEW
        IF(DELNEW.LT.ZSDEL)DEL=ZSDEL
```

```
      XAMXL=XLEFT-XLCLIP                                                80
      XBMXL=XRIGHT-XLCLIP
      XAMXR=XLEFT-XRCLIP
      XBMXR=XRIGHT-XRCLIP
      ZAL=ZLEFT
      ZBL=ZRIGHT
      ZAR=ZLEFT
      ZBR=ZRIGHT
      IF(ADJNEW)GO TO 320
      ZBL=ZLEFT
      ZAL=ZRIGHT
      ZBR=ZLEFT
      ZAR=ZRIGHT
320   XCMXL=ZS(2)-XLCLIP
      XDMXL=ZS(3)-XLCLIP
      XCMXR=ZS(2)-XRCLIP
      XDMXR=ZS(3)-XRCLIP
      IF(ZS(1)-2.GE.0)GO TO 321
      ZCL=ZS(4)
      ZDL=ZS(5)
      ZCR=ZS(4)
      ZDR=ZS(5)
      ADJOLD=ZSADJ
      GO TO 323
321   ADJOLD=.NOT.ADJNEW
      IF(ADJNEW)GO TO 322
      ZCL=ZS(4)
      ZDL=ZS(4)
      ZCR=ZS(5)
      ZDR=ZS(5)
      GO TO 323
322   ZCL=ZS(5)
      ZDL=ZS(5)
      ZCR=ZS(4)
      ZDR=ZS(4)
```

```
C       CLIP STATE *** ONE ADD TIME EACH PASS.
323     ABBCKL=.FALSE.
        ABBCKR=.FALSE.
        CDBCKL=.FALSE.
        CDBCKR=.FALSE.
        DELZ=.FALSE.
        NUMADD=NUMADD+1
        XHOLDL=(XAMXL+XBMXL)/2
        ZHOLDL=(ZAL+ZBL)/2
        XHOLDR=(XAMXR+XBMXR)/2
        ZHOLDR=(ZAR+ZBR)/2
        XTEMPL=(XCMXL+XDMXL)/2
        ZTEMPL=(ZCL+ZDL)/2
        XTEMPR=(XCMXR+XDMXR)/2
        ZTEMPR=(ZCR+ZDR)/2
        DEL=DEL/2
        IF(ZAL-ZDL.GE.0)ABBCKL=.TRUE.
        IF(ZCL-ZBL.GE.0)CDBCKL=.TRUE.
        IF(ZAR-ZDR.GE.0)ABBCKR=.TRUE.
        IF(ZCR-ZBR.GE.0)CDBCKR=.TRUE.
        IF(DEL.EQ.0)DELZ=.TRUE.
        LOG=((.NOT.ABLGE.OR..NOT.ABRLE).AND.((CDBCKL.AND..NOT.ABBCKR
       1.AND..NOT.CDBCKR).OR.(.NOT.ABBCKL.AND..NOT.CDBCKL.AND.CDBCKR)
       2.OR.(.NOT.ABBCKL.AND..NOT.CDBCKL.AND..NOT.ABBCKR)))
        LOG=LOG.OR.((.NOT.ABLLE.OR..NOT.ABRGE).AND.((ABBCKL.AND..NOT.
       1ABBCKR.AND..NOT.CDBCKR).OR.(.NOT.ABBCKL.AND..NOT.CDBCKL.AND.
       2ABBCKR).OR.(.NOT.ABBCKL.AND..NOT.CDBCKL.AND..NOT.CDBCKR)))
        LOG=LOG.OR.((.NOT.(ABBCKL.AND.ABBCKR).AND..NOT.(CDBCKL.AND.
       1CDBCKR)).AND.(ABLGE.AND.ABRLE.AND.ABLLE.AND.ABRGE))
        JCLIP=LOG.AND..NOT.DELZ
        LOG=((ABLGE.AND.ABRLE).AND.((ABBCKL.AND.ABBCKR).OR.(ABBCKL.AND.
       1.NOT.CDBCKR.AND.DELZ).OR.(ABBCKR.AND..NOT.CDBCKL.AND.DELZ)
       2.OR.(.NOT.CDBCKL.AND..NOT.CDBCKR.AND.DELZ)))
        J0BOX=LOG.OR.(DELZ.AND..NOT.ABBCKL.AND..NOT.CDBCKL
       1.AND..NOT.ABBCKR.AND..NOT.CDBCKR.AND.ABLGE.AND.ABRLE)
        LOG=(ABLLE.AND.ABRGE).AND..NOT.(ABLGE.AND.ABRLE.AND.((ABBCKL.
       1AND.ABBCKR).OR.(ABBCKL.AND..NOT.CDBCKR).OR.(ABBCKR
       2.AND..NOT.CDBCKL)))
        LOG=LOG.AND.((CDBCKL.AND.CDBCKR).OR.(CDBCKL.AND.
       1.NOT.ABBCKR.AND.DELZ).OR.(CDBCKR.AND..NOT.ABBCKL.AND.DELZ))
        J1BOX=LOG.OR.(DELZ.AND..NOT.ABBCKL.AND..NOT.CDBCKL
       1.AND..NOT.ABBCKR.AND..NOT.CDBCKR.AND.((ABLLE
       2.AND..NOT.ABRLE).OR.(.NOT.ABLGE.AND.ABRGE)))
        LOG=(DELZ.AND.((ABBCKL.AND..NOT.CDBCKL.AND..NOT.ABBCKR.AND.
       1CDBCKR).OR.(.NOT.ABBCKL.AND.CDBCKL.AND.ABBCKR
       2.AND..NOT.CDBCKR)))
        LOG=LOG.OR.((.NOT.ABLGE.OR..NOT.ABRLE).AND.((ABBCKL
       1.AND..NOT.CDBCKL).OR.(ABBCKR.AND..NOT.CDBCKR)))
        LOG=LOG.OR.((.NOT.ABLLE.OR..NOT.ABRGE).AND.((CDBCKL
       1.AND..NOT.ABBCKL).OR.(CDBCKR.AND..NOT.ABBCKR)))
        JBOXES=LOG.OR.(DELZ.AND..NOT.ABBCKL.AND..NOT.CDBCKL
       1.AND..NOT.ABBCKR.AND..NOT.CDBCKR.AND.((.NOT.ABLLE
       2.AND..NOT.ABRLE).OR.(.NOT.ABLGE.AND..NOT.ABRGE)))
        NUMCNT=0
        IF(JCLIP)NUMCNT=NUMCNT+1
        IF(J1BOX)NUMCNT=NUMCNT+1
        IF(JBOXES)NUMCNT=NUMCNT+1
        IF(J0BOX)NUMCNT=NUMCNT+1
        IF(NUMCNT.NE.1)PAUSE
        IF(JCLIP)GO TO 325
        IF(J1BOX)GO TO 331
        IF(JBOXES)GO TO 329
        IF(J0BOX)GO TO 335
```

```
325     IF(XHOLDL.GE.0)XBMXL=XHOLDL
        IF(XHOLDL.GE.0.AND.ADJNEW)ZBL=ZHOLDL
        IF(XHOLDL.GE.0.AND.(.NOT.ADJNEW))ZAL=ZHOLDL
        IF(XHOLDL.LT.0)XAMXL=XHOLDL
        IF(XHOLDL.LT.0.AND.ADJNEW)ZAL=ZHOLDL
        IF(XHOLDL.LT.0.AND.(.NOT.ADJNEW))ZBL=ZHOLDL
        IF(XHOLDR.GE.0)XBMXR=XHOLDR
        IF(XHOLDR.GE.0.AND.ADJNEW)ZBR=ZHOLDR
        IF(XHOLDR.GE.0.AND.(.NOT.ADJNEW))ZAR=ZHOLDR
        IF(XHOLDR.LT.0)XAMXR=XHOLDR
        IF(XHOLDR.LT.0.AND.ADJNEW)ZAR=ZHOLDR
        IF(XHOLDR.LT.0.AND.(.NOT.ADJNEW))ZBR=ZHOLDR
        IF(XTEMPL.GE.0)XDMXL=XTEMPL
        IF(XTEMPL.GE.0.AND.ADJOLD)ZDL=ZTEMPL
        IF(XTEMPL.GE.0.AND.(.NOT.ADJOLD))ZCL=ZTEMPL
        IF(XTEMPL.LT.0)XCMXL=XTEMPL
        IF(XTEMPL.LT.0.AND.ADJOLD)ZCL=ZTEMPL
        IF(XTEMPL.LT.0.AND.(.NOT.ADJOLD))ZDL=ZTEMPL
        IF(XTEMPR.GE.0)XDMXR=XTEMPR
        IF(XTEMPR.GE.0.AND.ADJOLD)ZDR=ZTEMPR
        IF(XTEMPR.GE.0.AND.(.NOT.ADJOLD))ZCR=ZTEMPR
        IF(XTEMPR.LT.0)XCMXR=XTEMPR
        IF(XTEMPR.LT.0.AND.ADJOLD)ZCR=ZTEMPR
        IF(XTEMPR.LT.0.AND.(.NOT.ADJOLD))ZDR=ZTEMPR
        GO TO 323
326     DEL=ZSDEL
        XAMXL=XLCLIP
        XBMXL=XRCLIP
        ZAL=ZAL-ZCL
        ZBL=ZAR-ZCR
327     ZHOLDL=(ZAL+ZBL)/2
        XHOLDL=(XAMXL+XBMXL)/2
        NUMADD=NUMADD+1
        DEL=DEL/2
        IF(DEL.EQ.0)GO TO 335
        IF(ZAL.XOR.ZHOLDL.GE.0)XAMXL=XHOLDL
        IF(ZBL.XOR.ZHOLDL.GE.0)XBMXL=XHOLDL
        IF(ZAL.XOR.ZHOLDL.GE.0)ZAL=ZHOLDL
        IF(ZBL.XOR.ZHOLDL.GE.0)ZBL=ZHOLDL
        GO TO 327
```

```
C       EXPAND BOX TO INCLUDE OLD BOX AND NEW LINE CLIPPED.
329     ZS(1)=ZS(1)+1
        IF(.NOT.DEPTH)GO TO 326
        IF(ABRLE.AND.ABRGE.AND.ABLLE.AND.ABLGE)GO TO 3295
        IF(ZLEFT-ZS(4).LT.0.AND.ADJNEW)ZS(4)=ZLEFT
        IF(ZRIGHT-ZS(4).LT.0.AND.(.NOT.ADJNEW))ZS(4)=ZRIGHT
        IF(ZLEFT-ZS(5).GE.0.AND.(.NOT.ADJNEW))ZS(5)=ZLEFT
        IF(ZRIGHT-ZS(5).GE.0.AND.ADJNEW)ZS(5)=ZRIGHT
        GO TO 330
3295    IF(ADJNEW.AND.ADJOLD.AND.CDBCKL)ZS(4)=ZAL
        IF(ADJNEW.AND.ADJOLD.AND.ABBCKL)ZS(4)=ZCL
        IF(ADJNEW.AND..NOT.ADJOLD.AND.ZAL.LT.ZCR)ZS(4)=ZAL
        IF(ADJNEW.AND..NOT.ADJOLD.AND.ZAL.GE.ZCR)ZS(4)=ZCR
        IF(.NOT.ADJNEW.AND.ADJOLD.AND.ZAR.LT.ZCL)ZS(4)=ZAR
        IF(.NOT.ADJNEW.AND.ADJOLD.AND.ZAR.GE.ZCL)ZS(4)=ZCL
        IF(.NOT.ADJNEW.AND..NOT.ADJOLD.AND.CDBCKR)ZS(4)=ZAR
        IF(.NOT.ADJNEW.AND..NOT.ADJOLD.AND.ABBCKR)ZS(4)=ZCR
        IF(ADJNEW.AND.ADJOLD.AND.CDBCKR)ZS(5)=ZDR
        IF(ADJNEW.AND.ADJOLD.AND.ABBCKR)ZS(5)=ZBR
        IF(ADJNEW.AND..NOT.ADJOLD.AND.ZBR.LT.ZDL)ZS(5)=ZDL
        IF(ADJNEW.AND..NOT.ADJOLD.AND.ZBR.GE.ZDL)ZS(5)=ZBR
        IF(.NOT.ADJNEW.AND.ADJOLD.AND.ZBL.LT.ZDR)ZS(5)=ZDR
        IF(.NOT.ADJNEW.AND.ADJOLD.AND.ZBL.GE.ZDR)ZS(5)=ZBL
        IF(.NOT.ADJNEW.AND..NOT.ADJOLD.AND.CDBCKL)ZS(5)=ZDL
        IF(.NOT.ADJNEW.AND..NOT.ADJOLD.AND.ABBCKL)ZS(5)=ZBL
330     IF(ABLLE)ZS(6)=XLTEST
        IF(ABRGE)ZS(7)=XRTEST
        IF(IXLEFT-ZS(8).LT.0)ZS(8)=IXLEFT
        ZSDEL=0
        ZS(10)=SEGPT
        IF(ABBCKL)GO TO 335
        ZS(10)=ZS(9)
        ZS(9)=SEGPT
        GO TO 335
C       MAKE A  ONE ELEMENT BOX.
331     ZS(1)=1
        ZS(2)=XLEFT
        ZS(3)=XRIGHT
        IF(ADJNEW)ZS(4)=ZLEFT
        IF(.NOT.ADJNEW)ZS(4)=ZRIGHT
        IF(ADJNEW)ZS(5)=ZRIGHT
        IF(.NOT.ADJNEW)ZS(5)=ZLEFT
        ZS(6)=XLTEST
        ZS(7)=XRTEST
        ZS(8)=IXLEFT
        ZS(9)=SEGPT
        ZSADJ=ADJNEW
        ZSDEL=DELNEW
335     CONTINUE
        NUMREF=(NUMADD+1)/2-Q(1,13)-Q(1,19)-Q(1,20)-NUMREF
        IF(NUMREF.GT.0)Q(1,14)=Q(1,14)+NUMREF
        IF(NUMADD.GT.20)NUMADD=20
        IF(NUMADD.LE.0)NUMADD=1
       ADDS(NUMADD)=ADDS(NUMADD)+1
        IF(.NOT.DEPTH)GO TO 402
       SEGPT=NEXT
        IF(FROM.EQ.0)GO TO 301
        GO TO 304
```

```
C       INTEREGATE THE ZS BOX
350     CONTINUE
        Q(1,13)=Q(1,13)+1
        IF(ZS(1)-2.LT.0)GO TO 355
        IF(SAM(1,1)-SAM(1,2).EQ.0)PAUSE 'SINGLE'
        IF(ZS(1).EQ.2.AND.(ZS(8).GE.SAM(1,2)))GO TO 400
C       SUBDIVISION NECESSARY.
        Q(1,4)=Q(1,4)+1
        IF(SEGOUT.EQ.0)GO TO 351
        SEG(SEGLO+11)=SEGACT
        SEGACT=SEGOUT
        Q(1,20)=Q(1,20)+1
        Q(1,19)=Q(1,19)+1
351     IF(ZS(8)-SAM(1,2).LT.0)GO TO 353
C       SUBDIVIDE IN THE MIDDLE.
        Q(1,12)=Q(1,12)+1
        SAM(1,2)=(SAM(1,1)+SAM(1,2))/2
        GO TO 299
C       SUBDIVIDE AT IXLEFT.
353     SAM(1,2)=ZS(8)
        GO TO 299
```

```
C       OUTPUT SEGMENTS.
355     IF(ZS(1).GT.0)GO TO 358
        SAM(2,1)=0
        Q(1,8)=Q(1,8)+1
356     XEND=SAM(1,2)
        POLYPT=0
        NEXTGO=1
        GO TO 368
358     CALL LDLPT(XEND,ZS(6))
        IF(XEND.EQ.SAM(1,1))GO TO 360
        POLYPT=0
        NEXTGO=2
        GO TO 368
360     CALL LDLPT(XEND,ZS(7))
        POLYPT=ZS(9)
        CALL LDLPT(POLYPT,SEG(POLYPT+1))
        CALL LDLPT(XTEMP,ZS(6))
        PQ(9)=PQ(9)+XEND-XTEMP+1
        NEXTGO=3
        GO TO 368
362     IF(XEND.EQ.SAM(1,2))GO TO 376
        GO TO 356
364     POLYPT=ZS(9)
        CALL LDLPT(POLYPT,SEG(POLYPT+1))
        PQ(9)=PQ(9)+SAM(1,2)-SAM(1,1)+1
        NEXTGO=4
        GO TO 368
365     XEND=SAM(1,2)
        POLYPT=ZS(10)
        CALL LDLPT(POLYPT,SEG(POLYPT+1))
        NEXTGO=1
        IF(FM.EQ.0)GO TO 368
        SAM(2,1)=IX
        SAM(3,1)=SEGPT+12
C       OUTPUT A SPECIFIC SEGMENT.
368     IF(SEGCNT.EQ.0)GO TO 372
        IF(POLYPT.NE.PRESEG)GO TO 372
        SAM(2,1)=0
        Q(1,8)=Q(1,8)+1
        GO TO 374
372     SEGCNT=SEGCNT+1
        Q(1,10)=Q(1,10)+1
        PRESEG=POLYPT
374     VISSEG(SEGCNT)=XEND
        CALL STLPT(POLYPT,VISSEG(SEGCNT))
        IF(SAM(2,1).EQ.0)GO TO 3755
C       STORE A SAMPLE POINT.
        Q(1,3)=Q(1,3)+1
        IF(SAM2S.NE.0)GO TO 375
        SAM2S=SAM(3,1)
        SAM2X=SAM(2,1)
        SAM2LX=SAM(2,1)
        SAM2L=SAM(3,1)
        GO TO 3755
375     IF(SAM(2,1).LE.SAM2LX)GO TO 3755
        SAM2LX=SAM(2,1)
        CALL STRPT(SAM(3,1),SEG(SAM2L))
        CALL STLPT(SAM(2,1),SEG(SAM2L))
        SAM2L=SAM(3,1)
3755    SAM(2,1)=0
        GO TO (376,360,362,365),NEXTGO
376     IF(SAM(1,2).EQ.FRAMEX)GO TO 498
        GO TO 281
```

```
C       INTERSECTING PLANES CASE.
400     DEPTH=.FALSE.
        FM=0
        ZS(1)=0
        Q(1,11)=Q(1,11)+1
        SEGPT=ZS(9)
        NEXT=-1
401     XLEFT=SEG(SEGPT+3)
        XRIGHT=SEG(SEGPT+5)
        ZLEFT=SEG(SEGPT+7)
        ZRIGHT=SEG(SEGPT+9)
        NUMREF=-Q(1,13)-Q(1,19)-Q(1,20)
        Q(1,19)=Q(1,19)+1
        GO TO 317
402     NEXT=NEXT+1
        SEGPT=ZS(10)
        IF(NEXT.EQ.0)GO TO 401
        DEPTH=.TRUE.
        XXTEST=XAMXL
        CALL LDLPT(XEND,XXTEST)
        Q(1,19)=Q(1,19)+2
        IF(IY.EQ.FRAMEY)GO TO 364
        SEGSAM=ZS(10)
        CALL STLPT(ZS(9),SEGSAM)
        CALL LDLPT(SEGPT,IMPLST)
        PREV=0
4010    IF(SEGPT.EQ.0)GO TO 4030
        Q(1,19)=Q(1,19)+1
        NEXT=SEG(SEGPT)
        IF(SEGSAM.NE.SEG(SEGPT+1)) GO TO 4020
        IF(PREV.EQ.0)CALL STLPT(NEXT,IMPLST)
        IF(PREV.NE.0)SEG(PREV)=NEXT
        SEG(SEGPT+4)=XXTEST-SEG(SEGPT+3)
        SEG(SEGPT+3)=XXTEST
        CALL LDLPT(IX,SEG(SEGPT+3)+SEG(SEGPT+4))
        IF(IX.LE.0.OR.IX.GT.FRAMEX)GO TO 4040
        FM=1
        GO TO 3091
4020    PREV=SEGPT
        SEGPT=NEXT
        GO TO 4010
4030    IF(IY.EQ.FRAMEY-1)GO TO 364
        CALL GETBLK(SEGPT)
        Q(1,23)=Q(1,23)+1
        PQ(1)=PQ(1)+1
        PQ(3)=PQ(3)+1
        IF(PQ(3).GT.PQ(2))PQ(2)=PQ(3)
        SEG(SEGPT+1)=SEGSAM
        SEG(SEGPT+2)=IY-FRAMEY+1
        SEG(SEGPT+11)=-1
        SEG(SEGPT+3)=XXTEST
        CALL LDRPT(SEG(SEGPT),IMPLST)
        CALL STRPT(SEGPT,IMPLST)
        GO TO 364
4040    CALL RETBLK(SEGPT)
        PQ(3)=PQ(3)-1
        Q(1,24)=Q(1,24)+1
        GO TO 364
```

```
498     CONTINUE
        DO 499 I=1,QL
        Q(10,I)=Q(10,I)+Q(1,I)
        Q(3,I)=Q(1,I)-Q(3,I)
        IF(Q(2,I).LT.Q(3,I))Q(2,I)=Q(3,I)
        Q(8,I)=Q(1,I)-Q(8,I)
499     IF(Q(7,I).LT.Q(8,I))Q(7,I)=Q(8,I)
        IF(IY.GT.FRAMEY)GO TO 500
        IF(PIX.NE.0)CALL SHOW
        IF(SEGCNT.NE.1)PQ(10)=PQ(10)+1
        GO TO 204
500     CONTINUE
        DO 501 I=1,QL
        Q(3,I)=Q(1,I)/FRAMEY
        Q(6,I)=10.**9/(30.*Q(1,I))
501     Q(8,I)=Q(1,I)/PQ(11)
        DO 502 I=13,QL
        PQ(11)=PQ(11)+Q(9,I)
502     PQ(13)=PQ(13)+Q(10,I)
        PQ(12)=10.**9/(30.*PQ(11))
        PQ(14)=10.**9/(30.*PQ(13))
        PQ(15)=PQ(11)+PQ(13)
        PQ(16)=10.**9/(30.*PQ(15))
        IF(STAT.EQ.0)RETURN
        TYPE 5002,(J,ADDS(J),J=1,20)
        TYPE 5001,(J,PQ(J),J=1,PQL)
        TYPE 5000,(J,(Q(I,J),I=1,10),J=1,QL)
        RETURN
5001    FORMAT(' PQ(',I2,')=',I6,/)
5000    FORMAT(' Q(X,',I2,')=',I6,4I4,1X,I9,2I4,2I6,/)
5002    FORMAT(' ADDS',/,5(' (',I2,')=',I6),/)
        END
```

APPENDIX II

STATISTICS OF OBJECTS AND ALGORITHMS

At the beginning of each set of statistics for a particular program there is a description of each of the counters. Following each description, a set of statistics for each of the ten test objects is compiled.

PQ(1)=NUMBER OF TOTAL BLOCKS (EDGE+POLY) REQUIRED FOR
HIDDEN LINE WORK.
PQ(2)=MAXIMUM NUMBER OF TOTAL BLOCKS EVER USED AT ONE TIME.

PQ(3)=CURRENT NUMBER OF TOTAL BLOCKS AT A GIVEN TIME.
(USED FOR CALCULATING PQ(2).)
PQ(4)=NUMBER OF EDGE BLOCKS REQUIRED FOR HIDDEN LINE WORK.

PQ(5)=MAXIMUM NUMBER OF EDGE BLOCKS EVER USED AT ONE TIME.

PQ(6)=CURRENT NUMBER OF EDGE BLOCKS AT A GIVEN TIME.
(USED FOR CALCULATING PQ(5).)
PQ(7)=NUMBER OF POLY BLOCKS REQUIRED FOR HIDDEN LINE WORK.

PQ(8)=MAXIMUM NUMBER OF POLY BLOCKS EVER USED AT ONE TIME.

PQ(9)=CURRENT NUMBER OF POLY BLOCKS AT A GIVEN TIME.
(USED FOR CALCULATING PQ(8).)
PQ(10)=TOTAL NUMBER OF EDGE BLOCKS IN FRAME.

PQ(11)=NUMBER OF EDGE BLOCKS WITH AT LEAST ONE OF THE
CONNECTED POLYGONS DRAWN CLOCKWISE.
PQ(12)=NUMBER OF THOSE EDGE BLOCKS OF PQ(11) WHOSE Y VALUE
OF THE BEGIN PT IS NOT THE SAME AS THE END PT Y VALUE.
PQ(13)=TOTAL NUMBER OF POLYGON BLOCKS IN THE FRAME.

PQ(14)=NUMBER OF POLYGON BLOCKS DRAWN CLOCKWISE.

PQ(15)=POINT DENSITY.

PQ(16)=NUMBER OF INVOLVED SCAN LINES.

Q COUNTERS

Q(1,X)=TOTAL PER FRAME
Q(2,X)=MAXIMUM REQUIRED OF A SCAN LINE
Q(3,X)=AVERAGE OF TOTAL SCAN LINES. ALSO SCRATCH FOR Q(2,X)
Q(4,X)=REQUIRED FOR PRE-FRAME PROCESSING
Q(5,X)=MAXIMUM REQUIRED FOR SCAN PREPARATION PROCESSING
Q(6,X)=NANOSECONDS REQUIRED. ALSO SCRATCH FOR Q(5,X)
Q(7,X)=MAXIMUM REQUIRED FOR SCAN DEPTH PROCESSING
Q(8,X)=AVERAGE OF ACTIVE SCAN LINES. ALSO SCRATCH FOR Q(7,X)
Q(9,X)=TOTAL FOR SCAN PREPARATION PROCESSING
Q(10,X)=TOTAL FOR SCAN DEPTH PROCESSING

Q(X,1)=NUMBER OF SLOPE CALCULATIONS.
Q(X,2)=NUMBER OF INTERCEPT CALCULATIONS.
Q(X,3)=NUMBER OF SAMPLE POINTS STORED FOR NEXT SCAN LINE.
Q(X,4)=SUBDIVISIONS (NOT FROM INTERSECTING CASE).
Q(X,5)=CONFIRMATIONS (EXCLUDED ARE SIMPLE CHECKS).
Q(X,6)=DEPTH SAMPLES REQUIRED.
Q(X,7)=DEPTH SAMPLES OF Q(X,6) NOT SATISFIED BY MAX-MIN TEST.
Q(X,8)=SAMPLE POINTS DELETED.
Q(X,9)=DEPTH CHECKS REQUIRED IN CONFIRMATION ROUTINE.
Q(X,10)=DEPTH SAMPLES OF Q(X,9) NOT SATISFIED BY MAX-MIN TEST.
Q(X,11)=OUTPUT SEGMENTS.
Q(X,12)=INTERCEPT SUBDIVISIONS.
Q(X,13)=INTERCEPT CALCULATIONS.
Q(X,14)=INTERCEPT SUBDIVISION RESOLUTIONS REACHED.
Q(X,15)=READS FROM POLY.
Q(X,16)=WRITES TO POLY.
Q(X,17)=READS FROM EDGE.
Q(X,18)=WRITES TO EDGE.
Q(X,19)=READS FROM POLY2
Q(X,20)=WRITES TO POLY2
Q(X,21)=READS FROM EDG2
Q(X,22)=WRITES TO EDG2

THE FOLLOWING HAVE NOT BEEN IMPLEMENTED.

Q(X,23)=READS FROM BUCKX
Q(X,24)=WRITES TO BUCKX
Q(X,25)=READS FROM BUCKY
Q(X,26)=WRITES TO BUCKY
Q(X,27)=READS FROM CX
Q(X,28)=READS FROM CZ
Q(X,29)=READS FROM X
Q(X,30)=READS FROM Y
Q(X,31)=READS FROM Z
Q(X,32)=READS FROM FAST TEMPORARY MEMORY.
Q(X,33)=WRITES TO FAST TEMPORARY MEMORY.

PQ(1)= 9172
PQ(2)= 76
PQ(3)= 0
PQ(4)= 92
PQ(5)= 31
PQ(6)= 0
PQ(7)= 9080
PQ(8)= 46
PQ(9)= 0
PQ(10)= 105
PQ(11)= 74
PQ(12)= 73
PQ(13)= 49
PQ(14)= 23
PQ(15)=104777
PQ(16)= 387

PENETRATION OBJECT VSG1 CONTINUED

Q(X, 1)=	73	3	0	0	3	456621	0	0	0	0
Q(X, 2)=	6349	31	12	0	31	5250	0	16	6349	0
Q(X, 3)=	3484	16	6	0	0	9567	16	9	0	3484
Q(X, 4)=	130	4	0	0	0	256410	4	0	0	130
Q(X, 5)=	157	9	0	0	0	212314	9	0	0	157
Q(X, 6)=	17504	240	34	0	0	1904	240	45	0	17504
Q(X, 7)=	14137	161	27	0	0	2357	161	36	0	14137
Q(X, 8)=	148	7	0	0	0	225225	7	0	0	148
Q(X, 9)=	1007	52	1	0	0	33101	52	2	0	1007
Q(X,10)=	128	12	0	0	0	260416	12	0	0	128
Q(X,11)=	4015	17	7	0	0	8302	17	10	0	4015
Q(X,12)=	26	4	0	0	0	1282051	4	0	0	26
Q(X,13)=	19	2	0	0	0	1754385	2	0	0	19
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	18329	93	35	77	92	1818	2	47	18160	19
Q(X,16)=	9129	46	17	49	46	3651	0	23	9080	0
Q(X,17)=	178	3	0	105	3	187265	0	0	0	0
Q(X,18)=	129	3	0	73	0	258397	3	0	0	56
Q(X,19)=	65788	496	128	0	92	506	412	169	18160	47628
Q(X,20)=	44361	217	86	0	175	751	42	114	35834	8527
Q(X,21)=	6349	31	12	0	31	5250	0	16	6349	0
Q(X,22)=	9868	44	19	0	31	3377	16	25	6349	3446

PQ(1)= 15658
PQ(2)= 134
PQ(3)= 0
PQ(4)= 380
PQ(5)= 58
PQ(6)= 0
PQ(7)= 15278
PQ(8)= 76
PQ(9)= 0
PQ(10)= 480
PQ(11)= 380
PQ(12)= 380
PQ(13)= 200
PQ(14)= 110
PQ(15)=124747
PQ(16)= 361

Q(X, 1)=	380	6	0	0	6	87719	0	1	0	0
Q(X, 2)=	11416	58	22	0	58	2919	0	31	11416	0
Q(X, 3)=	6565	29	12	0	0	5077	29	18	0	6565
Q(X, 4)=	292	5	0	0	0	114155	5	0	0	292
Q(X, 5)=	122	14	0	0	0	273224	14	0	0	122
Q(X, 6)=	34490	216	67	0	0	966	216	95	0	34490
Q(X, 7)=	29471	178	57	0	0	1131	178	81	0	29471
Q(X, 8)=	122	14	0	0	0	273224	14	0	0	122
Q(X, 9)=	674	106	1	0	0	49455	106	1	0	674
Q(X,10)=	85	13	0	0	0	392156	13	0	0	85
Q(X,11)=	7077	30	13	0	0	4710	30	19	0	7077
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	31246	156	61	310	156	1066	0	86	30556	0
Q(X,16)=	15478	76	30	200	76	2153	0	42	15278	0
Q(X,17)=	860	6	1	480	6	38759	0	2	0	0
Q(X,18)=	601	5	1	380	0	55463	5	1	0	221
Q(X,19)=	117217	728	228	0	152	284	581	324	30556	86661
Q(X,20)=	75446	378	147	0	303	441	75	208	60622	14824
Q(X,21)=	11416	58	22	0	58	2919	0	31	11416	0
Q(X,22)=	18133	86	35	0	62	1838	29	50	11416	6337

PQ(1)= 9712
PQ(2)= 101
PQ(3)= 0
PQ(4)= 192
PQ(5)= 41
PQ(6)= 0
PQ(7)= 9520
PQ(8)= 60
PQ(9)= 0
PQ(10)= 210
PQ(11)= 155
PQ(12)= 153
PQ(13)= 100
PQ(14)= 50
PQ(15)= 53613
PQ(16)= 358

Q(X, 1)=	153	5	0	0	5	217864	0	0	0	0
Q(X, 2)=	6688	41	13	0	41	4984	0	18	6688	0
Q(X, 3)=	3858	23	7	0	0	8640	23	10	0	3858
Q(X, 4)=	197	6	0	0	0	169204	6	0	0	197
Q(X, 5)=	213	8	0	0	0	156494	8	0	0	213
Q(X, 6)=	14130	160	27	0	0	2359	160	39	0	14130
Q(X, 7)=	11970	110	23	0	0	2784	110	33	0	11970
Q(X, 8)=	175	8	0	0	0	190476	8	0	0	175
Q(X, 9)=	928	51	1	0	0	35919	51	2	0	928
Q(X,10)=	166	10	0	0	0	200803	10	0	0	166
Q(X,11)=	4409	24	8	0	0	7560	24	12	0	4409
Q(X,12)=	12	3	0	0	0	2777777	3	0	0	12
Q(X,13)=	39	3	0	0	0	854700	3	0	0	39
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	19370	123	37	138	123	1720	3	54	19040	39
Q(X,16)=	9620	60	18	100	60	3465	0	26	9520	0
Q(X,17)=	363	5	0	210	5	91827	0	1	0	0
Q(X,18)=	192	2	0	153	0	173611	2	0	0	39
Q(X,19)=	65712	415	128	0	120	507	307	183	19040	46672
Q(X,20)=	46173	293	90	0	236	721	57	128	37299	8874
Q(X,21)=	6688	41	13	0	41	4984	0	18	6688	0
Q(X,22)=	10702	57	20	0	44	3114	23	29	6688	3861

PQ(1)= 7208
PQ(2)= 100
PQ(3)= 0
PQ(4)= 50
PQ(5)= 50
PQ(6)= 0
PQ(7)= 7158
PQ(8)= 50
PQ(9)= 0
PQ(10)= 300
PQ(11)= 100
PQ(12)= 50
PQ(13)= 150
PQ(14)= 25
PQ(15)=131769
PQ(16)= 363

Q(X, 1)=	50	2	0	0	2	666666	0	0	0	0
Q(X, 2)=	7158	50	13	0	50	4656	0	19	7158	0
Q(X, 3)=	726	2	1	0	0	45913	2	2	0	726
Q(X, 4)=	2	2	0	0	0	16666666	2	0	0	2
Q(X, 5)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 6)=	7162	50	13	0	0	4654	50	19	0	7162
Q(X, 7)=	1344	6	2	0	0	24801	6	3	0	1344
Q(X, 8)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,11)=	1238	3	2	0	0	26925	3	3	0	1238
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	14516	102	28	150	102	2296	0	39	14316	0
Q(X,16)=	7308	50	14	150	50	4561	0	20	7158	0
Q(X,17)=	350	2	0	300	2	95238	0	0	0	0
Q(X,18)=	52	2	0	50	0	641025	2	0	0	2
Q(X,19)=	31536	208	61	0	100	1056	108	86	14316	17220
Q(X,20)=	31848	224	62	0	199	1046	25	87	28269	3579
Q(X,21)=	7158	50	13	0	50	4656	0	19	7158	0
Q(X,22)=	7932	54	15	0	52	4202	2	21	7158	724

```
PQ( 1)= 15625
PQ( 2)=   156
PQ( 3)=     0
PQ( 4)=   215
PQ( 5)=    67
PQ( 6)=     0
PQ( 7)= 15410
PQ( 8)=    90
PQ( 9)=     0
PQ(10)=   300
PQ(11)=   225
PQ(12)=   215
PQ(13)=   150
PQ(14)=    75
PQ(15)=172829
PQ(16)=   492
```

```
Q(X, 1)=   215   4   0   0   4    155038   0   0     0     0
Q(X, 2)= 11556  67  22   0  67      2884   0  23 11556     0
Q(X, 3)=  7759  40  15   0   0      4296  40  15     0  7759
Q(X, 4)=   226   4   0   0   0    147492   4   0     0   226
Q(X, 5)=     3   1   0   0   0  11111111   1   0     0     3
Q(X, 6)=1046781054 204   0   0       3181054 212     0104678
Q(X, 7)= 33881 213  66   0   0       983 213  68     0 33881
Q(X, 8)=     3   1   0   0   0  11111111   1   0     0     3
Q(X, 9)=     8   4   0   0   0   4166666   4   0     0     8
Q(X,10)=     0   0   0   0   0 000000000   0   0     0     0
Q(X,11)=  8271  41  16   0   0      4030  41  16     0  8271
Q(X,12)=     0   0   0   0   0 000000000   0   0     0     0
Q(X,13)=     0   0   0   0   0 000000000   0   0     0     0
Q(X,14)=     0   0   0   0   0 000000000   0   0     0     0
Q(X,15)= 31260 183  61 225 183      1066   0  63 30820     0
Q(X,16)= 15560  90  30 150  90      2142   0  31 15410     0
Q(X,17)=   515   4   1 300   4     64724   0   1     0     0
Q(X,18)=   357   3   0 215   0     93370   3   0     0   142
Q(X,19)=1915151513 374   0 180       1741341 389 30820160695
Q(X,20)= 74756 432 146   0 350       445  83 151 60401 14355
Q(X,21)= 11556  67  22   0  67      2884   0  23 11556     0
Q(X,22)= 19362 108  37   0  70      1721  40  39 11556  7591
```

PQ(1)= 12553
PQ(2)= 84
PQ(3)= 0
PQ(4)= 125
PQ(5)= 36
PQ(6)= 0
PQ(7)= 12428
PQ(8)= 48
PQ(9)= 0
PQ(10)= 201
PQ(11)= 125
PQ(12)= 125
PQ(13)= 100
PQ(14)= 50
PQ(15)= 99425
PQ(16)= 456

Q(X, 1)=	125	4	0	0	4	266666	0	0	0	0
Q(X, 2)=	9321	36	18	0	36	3576	0	20	9321	0
Q(X, 3)=	5121	18	10	0	0	6509	18	11	0	5121
Q(X, 4)=	212	4	0	0	0	157232	4	0	0	212
Q(X, 5)=	190	4	0	0	0	175438	4	0	0	190
Q(X, 6)=	29084	150	56	0	0	1146	150	63	0	29084
Q(X, 7)=	27179	142	53	0	0	1226	142	59	0	27179
Q(X, 8)=	190	4	0	0	0	175438	4	0	0	190
Q(X, 9)=	731	15	1	0	0	45599	15	1	0	731
Q(X,10)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,11)=	5633	19	11	0	0	5917	19	12	0	5633
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	25156	100	49	175	100	1325	0	55	24856	0
Q(X,16)=	12528	48	24	100	48	2660	0	27	12428	0
Q(X,17)=	326	4	0	201	4	102249	0	0	0	0
Q(X,18)=	231	4	0	125	0	144300	4	0	0	106
Q(X,19)=	96653	415	188	0	96	344	319	211	24856	71797
Q(X,20)=	60765	237	118	0	191	548	46	133	49136	11629
Q(X,21)=	9321	36	18	0	36	3576	0	20	9321	0
Q(X,22)=	14514	56	28	0	40	2296	18	31	9321	5068

```
PQ( 1)= 25715
PQ( 2)=   175
PQ( 3)=     0
PQ( 4)=   125
PQ( 5)=    75
PQ( 6)=     0
PQ( 7)= 25590
PQ( 8)=   100
PQ( 9)=     0
PQ(10)=   201
PQ(11)=   125
PQ(12)=   125
PQ(13)=   100
PQ(14)=    50
PQ(15)=102921
PQ(16)=   316
```

Q(X, 1)=	125	5	0	0	5	266666	0	0	0	0
Q(X, 2)=	20338	75	39	0	75	1638	0	64	20338	0
Q(X, 3)=	11530	51	22	0	0	2891	51	36	0	11530
Q(X, 4)=	221	10	0	0	0	150829	10	0	0	221
Q(X, 5)=	40	4	0	0	0	833333	4	0	0	40
Q(X, 6)=	53836	324	105	0	0	619	324	170	0	53836
Q(X, 7)=	50804	324	99	0	0	656	324	160	0	50804
Q(X, 8)=	40	4	0	0	0	833333	4	0	0	40
Q(X, 9)=	40	4	0	0	0	833333	4	0	0	40
Q(X,10)=	40	4	0	0	0	833333	4	0	0	40
Q(X,11)=	12042	52	23	0	0	2768	52	38	0	12042
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	51480	202	100	175	202	647	0	162	51180	0
Q(X,16)=	25690	100	50	100	100	1297	0	81	25590	0
Q(X,17)=	326	5	0	201	5	102249	0	1	0	0
Q(X,18)=	173	4	0	125	0	192678	4	0	0	48
Q(X,19)=	196265	886	383	0	200	169	686	621	51180	145085
Q(X,20)=	126247	498	246	0	399	264	99	399	101401	24846
Q(X,21)=	20338	75	39	0	75	1638	0	64	20338	0
Q(X,22)=	31943	126	62	0	77	1043	51	101	20338	11480

PQ(1)= 12804
PQ(2)= 81
PQ(3)= 0
PQ(4)= 304
PQ(5)= 29
PQ(6)= 0
PQ(7)= 12500
PQ(8)= 56
PQ(9)= 0
PQ(10)= 308
PQ(11)= 306
PQ(12)= 304
PQ(13)= 192
PQ(14)= 190
PQ(15)= 60949
PQ(16)= 329

Q(X, 1)=	304	8	0	0	8	109649	0	0	0	0
Q(X, 2)=	6654	29	12	0	29	5009	0	20	6654	0
Q(X, 3)=	6464	27	12	0	0	5156	27	19	0	6464
Q(X, 4)=	322	8	0	0	0	103519	8	0	0	322
Q(X, 5)=	1	1	0	0	0	33333333	1	0	0	1
Q(X, 6)=	13680	98	26	0	0	2436	98	41	0	13680
Q(X, 7)=	13600	94	26	0	0	2450	94	41	0	13600
Q(X, 8)=	1	1	0	0	0	33333333	1	0	0	1
Q(X, 9)=	1	1	0	0	0	33333333	1	0	0	1
Q(X,10)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,11)=	6976	28	13	0	0	4778	28	21	0	6976
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	25537	116	49	233	116	1305	0	77	25000	0
Q(X,16)=	12692	56	24	192	56	2626	0	38	12500	0
Q(X,17)=	612	8	1	308	8	54466	0	1	0	0
Q(X,18)=	527	6	1	304	0	63251	6	1	0	223
Q(X,19)=	87771	388	171	0	112	379	284	266	25000	62771
Q(X,20)=	60917	258	118	0	211	547	51	185	49116	11801
Q(X,21)=	6654	29	12	0	29	5009	0	20	6654	0
Q(X,22)=	13185	60	25	0	35	2528	27	40	6654	6227

PQ(1)= 6319

PQ(2)= 52

PQ(3)= 0

PQ(4)= 147

PQ(5)= 20

PQ(6)= 0

PQ(7)= 6172

PQ(8)= 32

PQ(9)= 0

PQ(10)= 308

PQ(11)= 150

PQ(12)= 147

PQ(13)= 148

PQ(14)= 62

PQ(15)=113820

PQ(16)= 376

Q(X, 1)=	147	6	0	0	6	226757	0	0	0	0
Q(X, 2)=	3864	20	7	0	20	8626	0	10	3864	0
Q(X, 3)=	1550	6	3	0	0	21505	6	4	0	1550
Q(X, 4)=	126	3	0	0	0	264550	3	0	0	126
Q(X, 5)=	213	9	0	0	0	156494	9	0	0	213
Q(X, 6)=	8256	70	16	0	0	4037	70	21	0	8256
Q(X, 7)=	5799	45	11	0	0	5748	45	15	0	5799
Q(X, 8)=	120	9	0	0	0	277777	9	0	0	120
Q(X, 9)=	864	63	1	0	0	38580	63	2	0	864
Q(X,10)=	377	6	0	0	0	88417	6	1	0	377
Q(X,11)=	2062	7	4	0	0	16165	7	5	0	2062
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	12668	70	24	177	70	2631	0	33	12344	0
Q(X,16)=	6320	32	12	148	32	5274	0	16	6172	0
Q(X,17)=	455	6	0	308	6	73260	0	1	0	0
Q(X,18)=	171	2	0	147	0	194931	2	0	0	24
Q(X,19)=	36094	264	70	0	64	923	209	95	12344	23750
Q(X,20)=	29665	157	57	0	127	1123	30	78	24156	5509
Q(X,21)=	3864	20	7	0	20	8626	0	10	3864	0
Q(X,22)=	5540	32	10	0	26	6016	6	14	3864	1529

```
PQ( 1)=  8038
PQ( 2)=    73
PQ( 3)=     0
PQ( 4)=   168
PQ( 5)=    28
PQ( 6)=     0
PQ( 7)=  7870
PQ( 8)=    46
PQ( 9)=     0
PQ(10)=   308
PQ(11)=   150
PQ(12)=   145
PQ(13)=   148
PQ(14)=    62
PQ(15)=106613
PQ(16)=   363
```

Q(X, 1)=	145	5	0	0	5	229885	0	0	0	0
Q(X, 2)=	4717	28	9	0	28	7066	0	12	4717	0
Q(X, 3)=	2701	16	5	0	0	12341	16	7	0	2701
Q(X, 4)=	97	4	0	0	0	343642	4	0	0	97
Q(X, 5)=	195	9	0	0	0	170940	9	0	0	195
Q(X, 6)=	11998	104	23	0	0	2778	104	33	0	11998
Q(X, 7)=	10178	85	19	0	0	3275	85	28	0	10178
Q(X, 8)=	76	9	0	0	0	438596	9	0	0	76
Q(X, 9)=	681	63	1	0	0	48947	63	1	0	681
Q(X,10)=	250	4	0	0	0	133333	4	0	0	250
Q(X,11)=	3236	17	6	0	0	10300	17	8	0	3236
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	23	2	0	0	0	1449275	2	0	0	23
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	16085	96	31	177	96	2072	2	44	15740	23
Q(X,16)=	8018	46	15	148	46	4157	0	22	7870	0
Q(X,17)=	453	5	0	308	5	73583	0	1	0	0
Q(X,18)=	187	4	0	145	0	178253	4	0	0	42
Q(X,19)=	51714	354	101	0	92	644	265	142	15740	35974
Q(X,20)=	38134	228	74	0	183	874	45	105	30889	7245
Q(X,21)=	4717	28	9	0	28	7066	0	12	4717	0
Q(X,22)=	7552	45	14	0	31	4413	16	20	4717	2690

PQ(1)=NUMBER OF TOTAL BLOCKS REQUIRED FOR
HIDDEN LINE WORK.
PQ(2)=MAXIMUM NUMBER OF TOTAL BLOCKS EVER USED AT ONE TIME.

PQ(3)=CURRENT NUMBER OF TOTAL BLOCKS AT A GIVEN TIME.
(USED FOR CALCULATING PQ(2).)
PQ(4)=TOTAL NUMBER OF EDGE BLOCKS IN FRAME.

PQ(5)=NUMBER OF EDGE BLOCKS WITH AT LEAST ONE OF THE
CONNECTED POLYGONS DRAWN CLOCKWISE.
PQ(6)=NUMBER OF THOSE EDGE BLOCKS OF PQ(5) WHOSE Y VALUE
OF THE BEGIN PT IS NOT THE SAME AS THE END PT Y VALUE.
PQ(7)=TOTAL NUMBER OF POLYGON BLOCKS IN THE FRAME.

PQ(8)=NUMBER OF POLYGON BLOCKS DRAWN CLOCKWISE.

PQ(9)=POINT DENSITY.

PQ(10)=NUMBER OF INVOLVED SCAN LINES.

PQ(11)=MEMORY REFERENCES FOR X SORTER.

PQ(12)=NANOSECONDS PER MEMORY REFERENCE FOR X SORTER.

PQ(13)=MEMORY REFERENCES FOR DEPTH CALCULATOR.

PQ(14)=NANOSECONDS PER MEMORY REFERENCE FOR DEPTH CALCULATOR.

PQ(15)=MEMORY REF. (MAX PER SCAN LINE) OF PQ(11),PQ(13)

PQ(16)=NANOSECONDS FOR PQ(15).

PQ(1)= 112

PQ(2)= 44

PQ(3)= 0

PQ(4)= 105

PQ(5)= 74

PQ(6)= 73

PQ(7)= 49

PQ(8)= 23

PQ(9)=104777

PQ(10)= 387

PQ(11)= 904

PQ(12)= 918

PQ(13)= 853

Q(X, 1)=	73	3	0	0	3	456621	0	0	73	0
Q(X, 2)=	8718	43	17	0	43	3823	0	22	8718	0
Q(X, 3)=	3426	15	6	0	0	9729	15	8	0	3426
Q(X, 4)=	101	3	0	0	0	330033	3	0	0	101
Q(X, 5)=	315	14	0	0	0	105820	14	0	0	315
Q(X, 6)=	17402	230	33	0	0	1915	230	44	0	17402
Q(X, 7)=	13950	161	27	0	0	2389	161	36	0	13950
Q(X, 8)=	89	10	0	0	0	374531	10	0	0	89
Q(X, 9)=	1356	85	2	0	0	24582	85	3	0	1356
Q(X,10)=	440	17	0	0	0	75757	17	1	0	440
Q(X,11)=	3954	18	7	0	0	8430	18	10	0	3954
Q(X,12)=	22	5	0	0	0	1515151	5	0	0	22
Q(X,13)=	16	2	0	0	0	2083333	2	0	0	16
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	17512	157	34	77	43	1903	117	45	8718	8717
Q(X,16)=	8767	43	17	49	43	3802	0	22	8718	0
Q(X,17)=	178	3	0	105	3	187265	0	0	73	0
Q(X,18)=	73	0	0	73	0	456621	0	0	0	0
Q(X,19)=	46049	341	89	0	86	723	255	118	17548	28501
Q(X,20)=	29564	138	57	0	86	1127	55	76	17644	11920
Q(X,21)=	73133	418	142	0	176	455	254	188	36837	36296

PQ(1)= 520
PQ(2)= 79
PQ(3)= 0
PQ(4)= 480
PQ(5)= 380
PQ(6)= 380
PQ(7)= 200
PQ(8)= 110
PQ(9)=124747
PQ(10)= 361
PQ(11)= 515
PQ(12)= 496
PQ(13)= 481

Q(X, 1)=	380	6	0	0	6	87719	0	1	380	0
Q(X, 2)=	15278	76	29	0	76	2181	0	42	15278	0
Q(X, 3)=	6565	29	12	0	0	5077	29	18	0	6565
Q(X, 4)=	289	5	0	0	0	115340	5	0	0	289
Q(X, 5)=	119	14	0	0	0	280112	14	0	0	119
Q(X, 6)=	34486	216	67	0	0	966	216	95	0	34486
Q(X, 7)=	29487	178	57	0	0	1130	178	81	0	29487
Q(X, 8)=	119	14	0	0	0	280112	14	0	0	119
Q(X, 9)=	660	106	1	0	0	50505	106	1	0	660
Q(X,10)=	89	13	0	0	0	374531	13	0	0	89
Q(X,11)=	7077	30	13	0	0	4710	30	19	0	7077
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	32831	180	64	310	76	1015	108	90	15278	17243
Q(X,16)=	15478	76	30	200	76	2153	0	42	15278	0
Q(X,17)=	860	6	1	480	6	38759	0	2	380	0
Q(X,18)=	380	0	0	380	0	87719	0	1	0	0
Q(X,19)=	83160	484	162	0	152	400	343	230	31076	52084
Q(X,20)=	53353	257	104	0	160	624	102	147	31596	21757
Q(X,21)=	131814	638	257	0	324	252	326	365	64637	67177

PQ(1)= 246
PQ(2)= 63
PQ(3)= 0
PQ(4)= 210
PQ(5)= 155
PQ(6)= 153
PQ(7)= 100
PQ(8)= 50
PQ(9)= 53613
PQ(10)= 358
PQ(11)= 862
PQ(12)= 894
PQ(13)= 817

Q(X, 1)=	153	5	0	0	5	217864	0	0	153	0
Q(X, 2)=	9081	57	17	0	57	3670	0	25	9081	0
Q(X, 3)=	3841	22	7	0	0	8678	22	10	0	3841
Q(X, 4)=	182	5	0	0	0	183150	5	0	0	182
Q(X, 5)=	498	12	0	0	0	66934	12	1	0	498
Q(X, 6)=	13890	142	27	0	0	2399	142	38	0	13890
Q(X, 7)=	11789	93	23	0	0	2827	93	32	0	11789
Q(X, 8)=	184	10	0	0	0	181159	10	0	0	184
Q(X, 9)=	1473	67	2	0	0	22629	67	4	0	1473
Q(X,10)=	545	18	1	0	0	61162	18	1	0	545
Q(X,11)=	4392	24	8	0	0	7589	24	12	0	4392
Q(X,12)=	16	3	0	0	0	2083333	3	0	0	16
Q(X,13)=	38	3	0	0	0	877192	3	0	0	38
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	16202	124	31	138	57	2057	72	45	9081	6983
Q(X,16)=	9181	57	17	100	57	3630	0	25	9081	0
Q(X,17)=	363	5	0	210	5	91827	0	1	153	0
Q(X,18)=	153	0	0	153	0	217864	0	0	0	0
Q(X,19)=	47571	305	92	0	114	700	195	132	18408	29163
Q(X,20)=	31119	179	60	0	120	1071	68	86	18616	12503
Q(X,21)=	75923	433	148	0	244	439	223	212	38665	37258

PQ(1)= 50

PQ(2)= 50

PQ(3)= 0

PQ(4)= 300

PQ(5)= 100

PQ(6)= 50

PQ(7)= 150

PQ(8)= 25

PQ(9)=131769

PQ(10)= 363

PQ(11)= 1100

PQ(12)= 2165

PQ(13)= 1084

Q(X, 1)=	50	2	0	0	2	666666	0	0	50	0
Q(X, 2)=	7158	50	13	0	50	4656	0	19	7158	0
Q(X, 3)=	726	2	1	0	0	45913	2	2	0	726
Q(X, 4)=	2	2	0	0	0	16666666	2	0	0	2
Q(X, 5)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 6)=	7162	50	13	0	0	4654	50	19	0	7162
Q(X, 7)=	1344	6	2	0	0	24801	6	3	0	1344
Q(X, 8)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,11)=	1238	3	2	0	0	26925	3	3	0	1238
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	10889	75	21	150	50	3061	25	29	7158	3581
Q(X,16)=	7308	50	14	150	50	4561	0	20	7158	0
Q(X,17)=	350	2	0	300	2	95238	0	0	50	0
Q(X,18)=	50	0	0	50	0	666666	0	0	0	0
Q(X,19)=	24063	157	46	0	100	1385	57	66	14366	9697
Q(X,20)=	22300	154	43	0	102	1494	52	61	14416	7884
Q(X,21)=	45678	295	89	0	208	729	87	125	30285	15393

```
PQ( 1)=   282
PQ( 2)=    91
PQ( 3)=     0
PQ( 4)=   300
PQ( 5)=   225
PQ( 6)=   215
PQ( 7)=   150
PQ( 8)=    75
PQ( 9)=172829
PQ(10)=   492
PQ(11)=   514
PQ(12)=   310
PQ(13)=   310
```

```
Q(X, 1)=   215    4   0   0   4    155038   0   0   215      0
Q(X, 2)= 15410   90  30   0  90      2163   0  31 15410      0
Q(X, 3)=  7766   40  15   0   0      4292  40  15     0   7766
Q(X, 4)=   222    4   0   0   0    150150   4   0     0    222
Q(X, 5)=     4    1   0   0   0   8333333   1   0     0      4
Q(X, 6)=1046241052 204   0   0       3181052 212      0104624
Q(X, 7)= 33909 215  66   0   0       983 215  68      0 33909
Q(X, 8)=     4    1   0   0   0   8333333   1   0     0      4
Q(X, 9)=    11    4   0   0   0   3030303   4   0     0     11
Q(X,10)=     0    0   0   0   0 000000000   0   0     0      0
Q(X,11)=  8278   41  16   0   0      4026  41  16     0   8278
Q(X,12)=     0    0   0   0   0 000000000   0   0     0      0
Q(X,13)=     0    0   0   0   0 000000000   0   0     0      0
Q(X,14)=     0    0   0   0   0 000000000   0   0     0      0
Q(X,15)= 67947  612 132 225  90       490 526 138 15410  52312
Q(X,16)= 15560   90  30 150  90      2142   0  31 15410      0
Q(X,17)=   515    4   1 300   4     64724   0   1   215      0
Q(X,18)=   215    0   0 215   0    155038   0   0     0      0
Q(X,19)=121233  896 236   0 180       274 720 246 31102  90131
Q(X,20)= 54062  308 105   0 184       616 124 109 31384  22678
Q(X,21)=1720181170 335   0 372       193 806 349 64737107281
```

PQ(1)= 150
PQ(2)= 49
PQ(3)= 0
PQ(4)= 201
PQ(5)= 125
PQ(6)= 125
PQ(7)= 100
PQ(8)= 50
PQ(9)= 99425
PQ(10)= 456
PQ(11)= 639
PQ(12)= 591
PQ(13)= 586

Q(X, 1)=	125	4	0	0	4	266666	0	0	125	0
Q(X, 2)=	12428	48	24	0	48	2682	0	27	12428	0
Q(X, 3)=	5121	18	10	0	0	6509	18	11	0	5121
Q(X, 4)=	193	4	0	0	0	172711	4	0	0	193
Q(X, 5)=	152	3	0	0	0	219298	3	0	0	152
Q(X, 6)=	28938	142	56	0	0	1151	142	63	0	28938
Q(X, 7)=	26995	132	52	0	0	1234	132	59	0	26995
Q(X, 8)=	152	3	0	0	0	219298	3	0	0	152
Q(X, 9)=	541	15	1	0	0	61614	15	1	0	541
Q(X,10)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,11)=	5633	19	11	0	0	5917	19	12	0	5633
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	27072	119	52	175	48	1231	71	59	12428	14469
Q(X,16)=	12528	48	24	100	48	2660	0	27	12428	0
Q(X,17)=	326	4	0	201	4	102249	0	0	125	0
Q(X,18)=	125	0	0	125	0	266666	0	0	0	0
Q(X,19)=	68748	284	134	0	97	484	188	150	25006	43742
Q(X,20)=	42625	165	83	0	102	782	66	93	25156	17469
Q(X,21)=	108483	423	211	0	208	307	219	237	52137	56346

```
PQ( 1)=   150
PQ( 2)=   103
PQ( 3)=     0
PQ( 4)=   201
PQ( 5)=   125
PQ( 6)=   125
PQ( 7)=   100
PQ( 8)=    50
PQ( 9)=102921
PQ(10)=   316
PQ(11)=   319
PQ(12)=   284
PQ(13)=   275
```

Q(X, 1)=	125	5	0	0	5	266666	0	0	125	0
Q(X, 2)=	25590	100	49	0	100	1302	0	80	25590	0
Q(X, 3)=	11530	51	22	0	0	2891	51	36	0	11530
Q(X, 4)=	237	10	0	0	0	140646	10	0	0	237
Q(X, 5)=	68	4	0	0	0	490196	4	0	0	68
Q(X, 6)=	53938	324	105	0	0	617	324	170	0	53938
Q(X, 7)=	50981	324	99	0	0	653	324	161	0	50981
Q(X, 8)=	68	4	0	0	0	490196	4	0	0	68
Q(X, 9)=	160	9	0	0	0	208333	9	0	0	160
Q(X,10)=	68	4	0	0	0	490196	4	0	0	68
Q(X,11)=	12055	52	23	0	0	2765	52	38	0	12055
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	52734	260	102	175	100	632	162	166	25590	26969
Q(X,16)=	25690	100	50	100	100	1297	0	81	25590	0
Q(X,17)=	326	5	0	201	5	102249	0	1	125	0
Q(X,18)=	125	0	0	125	0	266666	0	0	0	0
Q(X,19)=	143144	617	279	0	201	232	417	452	51330	91814
Q(X,20)=	88170	351	172	0	204	378	151	279	51480	36690
Q(X,21)=	221483	926	432	0	412	150	522	700	104225	117258

PQ(1)= 566

PQ(2)= 62

PQ(3)= 0

PQ(4)= 308

PQ(5)= 306

PQ(6)= 304

PQ(7)= 192

PQ(8)= 190

PQ(9)= 60949

PQ(10)= 329

PQ(11)= 622

PQ(12)= 632

PQ(13)= 607

Q(X, 1)=	304	8	0	0	8	109649	0	0	304	0
Q(X, 2)=	12500	56	24	0	56	2666	0	37	12500	0
Q(X, 3)=	6463	27	12	0	0	5157	27	19	0	6463
Q(X, 4)=	318	8	0	0	0	104821	8	0	0	318
Q(X, 5)=	1	1	0	0	0	33333333	1	0	0	1
Q(X, 6)=	13656	98	26	0	0	2440	98	41	0	13656
Q(X, 7)=	13585	94	26	0	0	2453	94	41	0	13585
Q(X, 8)=	1	1	0	0	0	33333333	1	0	0	1
Q(X, 9)=	1	1	0	0	0	33333333	1	0	0	1
Q(X,10)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,11)=	6975	28	13	0	0	4778	28	21	0	6975
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	19561	96	38	233	56	1704	49	59	12500	6828
Q(X,16)=	12692	56	24	192	56	2626	0	38	12500	0
Q(X,17)=	612	8	1	308	8	54466	0	1	304	0
Q(X,18)=	304	0	0	304	0	109649	0	0	0	0
Q(X,19)=	63975	286	124	0	116	521	172	194	25566	38409
Q(X,20)=	44725	204	87	0	128	745	79	135	26132	18593
Q(X,21)=	106249	492	207	0	260	313	232	322	53581	52668

PQ(1)= 244

PQ(2)= 41

PQ(3)= 0

PQ(4)= 308

PQ(5)= 150

PQ(6)= 147

PQ(7)= 148

PQ(8)= 62

PQ(9)=113820

PQ(10)= 376

PQ(11)= 1226

PQ(12)= 1686

PQ(13)= 1196

Q(X, 1)=	147	6	0	0	6	226757	0	0	147	0
Q(X, 2)=	6172	32	12	0	32	5400	0	16	6172	0
Q(X, 3)=	1557	7	3	0	0	21408	7	4	0	1557
Q(X, 4)=	130	3	0	0	0	256410	3	0	0	130
Q(X, 5)=	226	9	0	0	0	147492	9	0	0	226
Q(X, 6)=	8308	80	16	0	0	4012	80	22	0	8308
Q(X, 7)=	5860	50	11	0	0	5688	50	15	0	5860
Q(X, 8)=	133	9	0	0	0	250626	9	0	0	133
Q(X, 9)=	936	63	1	0	0	35612	63	2	0	936
Q(X,10)=	381	6	0	0	0	87489	6	1	0	381
Q(X,11)=	2069	8	4	0	0	16110	8	5	0	2069
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	10503	72	20	177	32	3173	40	27	6172	4154
Q(X,16)=	6320	32	12	148	32	5274	0	16	6172	0
Q(X,17)=	455	6	0	308	6	73260	0	1	147	0
Q(X,18)=	147	0	0	147	0	226757	0	0	0	0
Q(X,19)=	26892	181	52	0	73	1239	123	71	12588	14304
Q(X,20)=	20457	120	39	0	82	1629	39	54	12832	7625
Q(X,21)=	46930	269	91	0	168	710	113	124	27169	19761

```
PQ( 1)=   262
PQ( 2)=    45
PQ( 3)=     0
PQ( 4)=   308
PQ( 5)=   150
PQ( 6)=   145
PQ( 7)=   148
PQ( 8)=    62
PQ( 9)=106613
PQ(10)=   363
PQ(11)=  1173
PQ(12)=  1333
PQ(13)=  1137
```

Q(X, 1)=	145	5	0	0	5	229885	0	0	145	0
Q(X, 2)=	6487	37	12	0	37	5138	0	17	6487	0
Q(X, 3)=	2281	14	4	0	0	14613	14	6	0	2281
Q(X, 4)=	122	6	0	0	0	273224	6	0	0	122
Q(X, 5)=	588	9	1	0	0	56689	9	1	0	588
Q(X, 6)=	10926	108	21	0	0	3050	108	30	0	10926
Q(X, 7)=	9001	93	17	0	0	3703	93	24	0	9001
Q(X, 8)=	91	9	0	0	0	366300	9	0	0	91
Q(X, 9)=	1297	63	2	0	0	25700	63	3	0	1297
Q(X,10)=	682	10	1	0	0	48875	10	1	0	682
Q(X,11)=	2813	15	5	0	0	11849	15	7	0	2813
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	20	2	0	0	0	1666666	2	0	0	20
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	12147	89	23	177	37	2744	55	33	6487	5483
Q(X,16)=	6635	37	12	148	37	5023	0	18	6487	0
Q(X,17)=	453	5	0	308	5	73583	0	1	145	0
Q(X,18)=	145	0	0	145	0	229885	0	0	0	0
Q(X,19)=	32663	208	63	0	81	1020	136	89	13236	19427
Q(X,20)=	21775	131	42	0	89	1530	46	59	13478	8297
Q(X,21)=	53403	339	104	0	182	624	163	147	28409	24994

PQ(1)=NUMBER OF TOTAL BLOCKS REQUIRED FOR
HIDDEN LINE WORK.
PQ(2)=MAXIMUM NUMBER OF TOTAL BLOCKS EVER USED AT ONE TIME.

PQ(3)=CURRENT NUMBER OF TOTAL BLOCKS AT A GIVEN TIME.
(USED FOR CALCULATING PQ(2).)
PQ(4)=TOTAL NUMBER OF EDGE BLOCKS IN FRAME.

PQ(5)=NUMBER OF EDGE BLOCKS WITH AT LEAST ONE OF THE
CONNECTED POLYGONS DRAWN CLOCKWISE.
PQ(6)=NUMBER OF THOSE EDGE BLOCKS OF PQ(5) WHOSE Y VALUE
OF THE BEGIN PT IS NOT THE SAME AS THE END PT Y VALUE.
PQ(7)=TOTAL NUMBER OF POLYGON BLOCKS IN THE FRAME.

PQ(8)=NUMBER OF POLYGON BLOCKS DRAWN CLOCKWISE.

PQ(9)=POINT DENSITY.

PQ(10)=NUMBER OF INVOLVED SCAN LINES.

PQ(11)=NANOSECONDS PER MEMORY REFERENCE FOR X SORTER.

PQ(12)=NANOSECONDS PER MEMORY REFERENCE FOR DEPTH CALCULATOR.

PQ(13)=NANOSECONDS PER MAXIMUM OF EACH PROCESSOR.

```
Q COUNTERS
Q(1,X)=TOTAL PER FRAME
Q(2,X)=MAXIMUM REQUIRED OF A SCAN LINE
Q(3,X)=AVERAGE OF TOTAL SCAN LINES. ALSO SCRATCH FOR Q(2,X)
Q(4,X)=REQUIRED FOR PRE-FRAME PROCESSING
Q(5,X)=MAXIMUM REQUIRED FOR SCAN PREPARATION PROCESSING
Q(6,X)=NANOSECONDS REQUIRED. ALSO SCRATCH FOR Q(5,X)
Q(7,X)=MAXIMUM REQUIRED FOR SCAN DEPTH PROCESSING
Q(8,X)=AVERAGE OF ACTIVE SCAN LINES. ALSO SCRATCH FOR Q(7,X)
Q(9,X)=TOTAL FOR SCAN PREPARATION PROCESSING
Q(10,X)=TOTAL FOR SCAN DEPTH PROCESSING


Q(X,1)=NUMBER OF SLOPE CALCULATIONS.
Q(X,2)=NUMBER OF INTERCEPT CALCULATIONS.
Q(X,3)=NUMBER OF SAMPLE POINTS STORED FOR NEXT SCAN LINE.
Q(X,4)=SUBDIVISIONS (NOT FROM INTERSECTING CASE).
Q(X,5)=CONFIRMATIONS (EXCLUDED ARE SIMPLE CHECKS).
Q(X,6)=DEPTH SAMPLES REQUIRED.
Q(X,7)=DEPTH SAMPLES OF Q(X,6) NOT SATISFIED BY MAX-MIN TEST.
Q(X,8)=SAMPLE POINTS DELETED.
Q(X,9)=DEPTH CHECKS REQUIRED IN CONFIRMATION ROUTINE.
Q(X,10)=DEPTH SAMPLES OF Q(X,9) NOT SATISFIED BY MAX-MIN TEST.
Q(X,11)=OUTPUT SEGMENTS.
Q(X,12)=INTERCEPT SUBDIVISIONS.
Q(X,13)=INTERCEPT CALCULATIONS.
Q(X,14)=INTERCEPT SUBDIVISION RESOLUTIONS REACHED.
Q(X,15)=READS FROM POLY.
Q(X,16)=WRITES TO POLY.
Q(X,17)=READS FROM EDGE.
Q(X,18)=WRITES TO EDGE.
Q(X,19)=READS FROM EDGE2
Q(X,20)=WRITES TO EDGE2
Q(X,21)=MEMORY CYCLES

THE FOLLOWING HAVE NOT BEEN IMPLEMENTED.

Q(X,23)=READS FROM BUCKX
Q(X,24)=WRITES TO BUCKX
Q(X,25)=READS FROM BUCKY
Q(X,26)=WRITES TO BUCKY
Q(X,27)=READS FROM CX
Q(X,28)=READS FROM CZ
Q(X,29)=READS FROM X
Q(X,30)=READS FROM Y
Q(X,31)=READS FROM Z
Q(X,32)=READS FROM FAST TEMPORARY MEMORY.
Q(X,33)=WRITES TO FAST TEMPORARY MEMORY.
```

PQ(1)= 38

PQ(2)= 25

PQ(3)= 0

PQ(4)= 105

PQ(5)= 74

PQ(6)= 73

PQ(7)= 49

PQ(8)= 23

PQ(9)=104777

PQ(10)= 387

PQ(11)= 34518

PQ(12)= 965

PQ(13)= 37181

PQ(14)= 896

PQ(15)= 37919

PQ(16)= 879

Q(X, 1)=	73	3	0	0	3	456621	0	0	73	0
Q(X, 2)=	4255	19	8	0	19	7833	0	10	4255	0
Q(X, 3)=	3415	15	6	0	0	9760	15	8	0	3415
Q(X, 4)=	68	3	0	0	0	490196	3	0	0	68
Q(X, 5)=	297	11	0	0	0	112233	11	0	0	297
Q(X, 6)=	17144	230	33	0	0	1944	230	44	0	17144
Q(X, 7)=	13677	160	26	0	0	2437	160	35	0	13677
Q(X, 8)=	69	7	0	0	0	483091	7	0	0	69
Q(X, 9)=	1571	66	3	0	0	21217	66	4	0	1571
Q(X,10)=	715	20	1	0	0	46620	20	1	0	715
Q(X,11)=	3942	18	7	0	0	8455	18	10	0	3942
Q(X,12)=	22	5	0	0	0	1515151	5	0	0	22
Q(X,13)=	15	2	0	0	0	2222222	2	0	0	15
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	388	9	0	77	9	85910	0	1	311	0
Q(X,16)=	310	8	0	49	8	107526	0	0	261	0
Q(X,17)=	178	3	0	105	3	187265	0	0	73	0
Q(X,18)=	73	0	0	73	0	456621	0	0	0	0
Q(X,19)=	34467	278	67	0	45	967	238	89	8912	25555
Q(X,20)=	16487	76	32	0	44	2021	36	42	8818	7669
Q(X,21)=	8408	38	16	0	38	3964	0	21	8408	0
Q(X,22)=	4239	19	8	0	19	7863	0	10	4239	0
Q(X,23)=	38	2	0	0	2	877192	2	0	23	15
Q(X,24)=	38	3	0	0	3	877192	0	0	38	0
Q(X,25)=	47	1	0	0	1	709219	0	0	47	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,27)=	3942	18	7	0	0	8455	18	10	0	3942
Q(X,28)=	3388	16	6	0	16	9838	0	8	3388	0

PQ(1)= 130

PQ(2)= 39

PQ(3)= 0

PQ(4)= 480

PQ(5)= 380

PQ(6)= 380

PQ(7)= 200

PQ(8)= 110

PQ(9)=124747

PQ(10)= 361

PQ(11)= 65218

PQ(12)= 511

PQ(13)= 67598

PQ(14)= 493

PQ(15)= 69483

PQ(16)= 479

Q(X, 1)=	380	6	0	0	6	87719	0	1	380	0
Q(X, 2)=	7639	38	14	0	38	4363	0	21	7639	0
Q(X, 3)=	6565	29	12	0	0	5077	29	18	0	6565
Q(X, 4)=	206	4	0	0	0	161812	4	0	0	206
Q(X, 5)=	122	14	0	0	0	273224	14	0	0	122
Q(X, 6)=	33702	202	65	0	0	989	202	93	0	33702
Q(X, 7)=	28823	171	56	0	0	1156	171	79	0	28823
Q(X, 8)=	122	14	0	0	0	273224	14	0	0	122
Q(X, 9)=	1124	119	2	0	0	29655	119	3	0	1124
Q(X,10)=	445	18	0	0	0	74906	18	1	0	445
Q(X,11)=	7077	30	13	0	0	4710	30	19	0	7077
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	1983	20	3	310	20	16809	0	5	1673	0
Q(X,16)=	1628	16	3	200	16	20475	0	4	1428	0
Q(X,17)=	860	6	1	480	6	38759	0	2	380	0
Q(X,18)=	380	0	0	380	0	87719	0	1	0	0
Q(X,19)=	62744	399	122	0	89	531	321	173	16384	46360
Q(X,20)=	29961	142	58	0	83	1112	64	82	15800	14161
Q(X,21)=	15117	76	29	0	76	2205	0	41	15117	0
Q(X,22)=	7596	38	14	0	38	4388	0	21	7596	0
Q(X,23)=	130	3	0	0	3	256410	0	0	130	0
Q(X,24)=	130	4	0	0	4	256410	0	0	130	0
Q(X,25)=	174	1	0	0	1	191570	0	0	174	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,27)=	7077	30	13	0	0	4710	30	19	0	7077
Q(X,28)=	6406	29	12	0	29	5203	0	17	6406	0

PQ(1)= 88
PQ(2)= 30
PQ(3)= 0
PQ(4)= 210
PQ(5)= 155
PQ(6)= 153
PQ(7)= 100
PQ(8)= 50
PQ(9)= 53613
PQ(10)= 358
PQ(11)= 37224
PQ(12)= 895
PQ(13)= 38895
PQ(14)= 857
PQ(15)= 40563
PQ(16)= 821

Q(X, 1)=	153	5	0	0	5	217864	0	0	153	0
Q(X, 2)=	4415	27	8	0	27	7550	0	12	4415	0
Q(X, 3)=	3942	23	7	0	0	8455	23	11	0	3942
Q(X, 4)=	148	7	0	0	0	225225	7	0	0	148
Q(X, 5)=	515	11	1	0	0	64724	11	1	0	515
Q(X, 6)=	13906	152	27	0	0	2397	152	38	0	13906
Q(X, 7)=	11669	114	22	0	0	2856	114	32	0	11669
Q(X, 8)=	199	9	0	0	0	167504	9	0	0	199
Q(X, 9)=	1959	63	3	0	0	17015	63	5	0	1959
Q(X,10)=	787	20	1	0	0	42354	20	2	0	787
Q(X,11)=	4492	25	8	0	0	7420	25	12	0	4492
Q(X,12)=	11	3	0	0	0	3030303	3	0	0	11
Q(X,13)=	38	3	0	0	0	877192	3	0	0	38
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	808	21	1	138	21	41254	0	2	670	0
Q(X,16)=	672	19	1	100	19	49603	0	1	572	0
Q(X,17)=	363	5	0	210	5	91827	0	1	153	0
Q(X,18)=	153	0	0	153	0	217864	0	0	0	0
Q(X,19)=	35617	247	69	0	71	935	187	99	9544	26073
Q(X,20)=	17629	95	34	0	60	1890	44	49	9337	8292
Q(X,21)=	8483	52	16	0	52	3929	0	23	8483	0
Q(X,22)=	4312	26	8	0	26	7730	0	12	4312	0
Q(X,23)=	88	4	0	0	3	378787	3	0	50	38
Q(X,24)=	88	4	0	0	4	378787	0	0	88	0
Q(X,25)=	83	1	0	0	1	401606	0	0	83	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,27)=	4492	25	8	0	0	7420	25	12	0	4492
Q(X,28)=	3932	24	7	0	24	8477	0	10	3932	0

PQ(1)= 25

PQ(2)= 25

PQ(3)= 0

PQ(4)= 300

PQ(5)= 100

PQ(6)= 50

PQ(7)= 150

PQ(8)= 25

PQ(9)=131769

PQ(10)= 363

PQ(11)= 26352

PQ(12)= 1264

PQ(13)= 17371

PQ(14)= 1918

PQ(15)= 26660

PQ(16)= 1250

Q(X, 1)=	50	2	0	0	2	666666	0	0	50	0
Q(X, 2)=	3579	25	6	0	25	9313	0	9	3579	0
Q(X, 3)=	726	2	1	0	0	45913	2	2	0	726
Q(X, 4)=	2	2	0	0	0	16666666	2	0	0	2
Q(X, 5)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 6)=	7162	50	13	0	0	4654	50	19	0	7162
Q(X, 7)=	1344	6	2	0	0	24801	6	3	0	1344
Q(X, 8)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,11)=	1238	3	2	0	0	26925	3	3	0	1238
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	325	5	0	150	5	102564	0	0	175	0
Q(X,16)=	275	3	0	150	3	121212	0	0	125	0
Q(X,17)=	350	2	0	300	2	95238	0	0	50	0
Q(X,18)=	50	0	0	50	0	666666	0	0	0	0
Q(X,19)=	19086	130	37	0	52	1746	78	52	7258	11828
Q(X,20)=	11513	79	22	0	52	2895	27	31	7208	4305
Q(X,21)=	7158	50	13	0	50	4656	0	19	7158	0
Q(X,22)=	3579	25	6	0	25	9313	0	9	3579	0
Q(X,23)=	25	1	0	0	1	1333333	0	0	25	0
Q(X,24)=	25	1	0	0	1	1333333	0	0	25	0
Q(X,25)=	25	1	0	0	1	1333333	0	0	25	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,27)=	1238	3	2	0	0	26925	3	3	0	1238
Q(X,28)=	724	2	1	0	2	46040	0	1	724	0

PQ(1)= 75

PQ(2)= 45

PQ(3)= Ø

PQ(4)= 300

PQ(5)= 225

PQ(6)= 215

PQ(7)= 150

PQ(8)= 75

PQ(9)=172829

PQ(10)= 492

PQ(11)= 63253

PQ(12)= 526

PQ(13)=105137

PQ(14)= 317

PQ(15)=105421

PQ(16)= 316

Q(X, 1)=	215	4	0	0	4	155038	0	0	215	0
Q(X, 2)=	7705	45	15	0	45	4326	0	15	7705	0
Q(X, 3)=	7766	40	15	0	0	4292	40	15	0	7766
Q(X, 4)=	168	4	0	0	0	198412	4	0	0	168
Q(X, 5)=	3	1	0	0	0	11111111	1	0	0	3
Q(X, 6)=	103384	990	201	0	0	322	990	210	0	103384
Q(X, 7)=	33535	203	65	0	0	993	203	68	0	33535
Q(X, 8)=	3	1	0	0	0	11111111	1	0	0	3
Q(X, 9)=	11	5	0	0	0	3030303	5	0	0	11
Q(X,10)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,11)=	8278	41	16	0	0	4026	41	16	0	8278
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	1116	14	2	225	14	29868	0	2	891	0
Q(X,16)=	893	13	1	150	13	37327	0	1	743	0
Q(X,17)=	515	4	1	300	4	64724	0	1	215	0
Q(X,18)=	215	0	0	215	0	155038	0	0	0	0
Q(X,19)=	97348	744	190	0	93	342	653	197	15711	81637
Q(X,20)=	30665	173	59	0	91	1087	82	62	15443	15222
Q(X,21)=	14895	84	29	0	84	2237	0	30	14895	0
Q(X,22)=	7456	42	14	0	42	4470	0	15	7456	0
Q(X,23)=	75	2	0	0	2	444444	0	0	75	0
Q(X,24)=	75	3	0	0	3	444444	0	0	75	0
Q(X,25)=	108	1	0	0	1	308641	0	0	108	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,27)=	8278	41	16	0	0	4026	41	16	0	8278
Q(X,28)=	7641	40	14	0	40	4362	0	15	7641	0

PQ(1)= 50

PQ(2)= 24

PQ(3)= 0

PQ(4)= 201

PQ(5)= 125

PQ(6)= 125

PQ(7)= 100

PQ(8)= 50

PQ(9)= 99425

PQ(10)= 456

PQ(11)= 49900

PQ(12)= 668

PQ(13)= 56041

PQ(14)= 594

PQ(15)= 56385

PQ(16)= 591

Q(X, 1)=	125	4	0	0	4	266666	0	0	125	0
Q(X, 2)=	6214	24	12	0	24	5364	0	13	6214	0
Q(X, 3)=	5121	18	10	0	0	6509	18	11	0	5121
Q(X, 4)=	127	3	0	0	0	262467	3	0	0	127
Q(X, 5)=	94	3	0	0	0	354609	3	0	0	94
Q(X, 6)=	28632	142	55	0	0	1164	142	62	0	28632
Q(X, 7)=	26698	130	52	0	0	1248	130	58	0	26698
Q(X, 8)=	94	3	0	0	0	354609	3	0	0	94
Q(X, 9)=	2685	19	5	0	0	12414	19	5	0	2685
Q(X,10)=	2371	11	4	0	0	14058	11	5	0	2371
Q(X,11)=	5633	19	11	0	0	5917	19	12	0	5633
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	625	10	1	175	10	53333	0	1	450	0
Q(X,16)=	500	9	0	100	9	66666	0	1	400	0
Q(X,17)=	326	4	0	201	4	102249	0	0	125	0
Q(X,18)=	125	0	0	125	0	266666	0	0	0	0
Q(X,19)=	51801	212	101	0	53	643	160	113	12688	39113
Q(X,20)=	23833	93	46	0	52	1398	42	52	12538	11295
Q(X,21)=	12256	48	23	0	48	2719	0	26	12256	0
Q(X,22)=	6174	24	12	0	24	5398	0	13	6174	0
Q(X,23)=	50	2	0	0	2	666666	0	0	50	0
Q(X,24)=	50	2	0	0	2	666666	0	0	50	0
Q(X,25)=	72	1	0	0	1	462962	0	0	72	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,27)=	5633	19	11	0	0	5917	19	12	0	5633
Q(X,28)=	5097	18	9	0	18	6539	0	11	5097	0

```
PQ( 1)=    50
PQ( 2)=    50
PQ( 3)=     0
PQ( 4)=   201
PQ( 5)=   125
PQ( 6)=   125
PQ( 7)=   100
PQ( 8)=    50
PQ( 9)=102921
PQ(10)=   314
PQ(11)=101279
PQ(12)=   329
PQ(13)=117105
PQ(14)=   284
PQ(15)=117373
PQ(16)=   283
```

Q(X, 1)=	125	5	0	0	5	266666	0	0	125	0
Q(X, 2)=	12795	50	24	0	50	2605	0	40	12795	0
Q(X, 3)=	11530	51	22	0	0	2891	51	36	0	11530
Q(X, 4)=	157	8	0	0	0	212314	8	0	0	157
Q(X, 5)=	43	2	0	0	0	775193	2	0	0	43
Q(X, 6)=	53332	282	104	0	0	625	282	169	0	53332
Q(X, 7)=	50307	282	98	0	0	662	282	160	0	50307
Q(X, 8)=	43	2	0	0	0	775193	2	0	0	43
Q(X, 9)=	6121	31	11	0	0	5445	31	19	0	6121
Q(X,10)=	6010	24	11	0	0	5546	24	19	0	6010
Q(X,11)=	12042	52	23	0	0	2768	52	38	0	12042
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	625	23	1	175	23	53333	0	1	450	0
Q(X,16)=	500	23	0	100	23	66666	0	1	400	0
Q(X,17)=	326	5	0	201	5	102249	0	1	125	0
Q(X,18)=	125	0	0	125	0	266666	0	0	0	0
Q(X,19)=	106628	463	208	0	100	312	363	339	25675	80953
Q(X,20)=	49635	201	96	0	100	671	101	158	25525	24110
Q(X,21)=	24905	100	48	0	100	1338	0	79	24905	0
Q(X,22)=	12580	50	24	0	50	2649	0	40	12580	0
Q(X,23)=	50	2	0	0	2	666666	0	0	50	0
Q(X,24)=	50	5	0	0	5	666666	0	0	50	0
Q(X,25)=	43	1	0	0	1	775193	0	0	43	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,27)=	12042	52	23	0	0	2768	52	38	0	12042
Q(X,28)=	11476	51	22	0	51	2904	0	36	11476	0

PQ(1)= 190
PQ(2)= 29
PQ(3)= 0
PQ(4)= 308
PQ(5)= 306
PQ(6)= 304
PQ(7)= 192
PQ(8)= 190
PQ(9)= 60949
PQ(10)= 329
PQ(11)= 54543
PQ(12)= 611
PQ(13)= 51055
PQ(14)= 652
PQ(15)= 55459
PQ(16)= 601

Q(X, 1)=	304	8	0	0	8	109649	0	0	304	0
Q(X, 2)=	6250	28	12	0	28	5333	0	18	6250	0
Q(X, 3)=	6463	27	12	0	0	5157	27	19	0	6463
Q(X, 4)=	256	6	0	0	0	130208	6	0	0	256
Q(X, 5)=	1	1	0	0	0	33333333	1	0	0	1
Q(X, 6)=	13218	78	25	0	0	2521	78	40	0	13218
Q(X, 7)=	13201	78	25	0	0	2525	78	40	0	13201
Q(X, 8)=	1	1	0	0	0	33333333	1	0	0	1
Q(X, 9)=	1	1	0	0	0	33333333	1	0	0	1
Q(X,10)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,11)=	6975	28	13	0	0	4778	28	21	0	6975
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	1744	28	3	233	28	19113	0	5	1511	0
Q(X,16)=	1700	28	3	192	28	19607	0	5	1508	0
Q(X,17)=	612	8	1	308	8	54466	0	1	304	0
Q(X,18)=	304	0	0	304	0	109649	0	0	0	0
Q(X,19)=	44999	220	87	0	75	740	145	136	13447	31552
Q(X,20)=	25409	116	49	0	65	1311	53	77	12881	12528
Q(X,21)=	12130	52	23	0	52	2748	0	36	12130	0
Q(X,22)=	6065	26	11	0	26	5496	0	18	6065	0
Q(X,23)=	190	6	0	0	6	175438	0	0	190	0
Q(X,24)=	190	6	0	0	6	175438	0	0	190	0
Q(X,25)=	100	1	0	0	1	333333	0	0	100	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,27)=	6975	28	13	0	0	4778	28	21	0	6975
Q(X,28)=	6217	27	12	0	27	5361	0	18	6217	0

PQ(1)=NUMBER OF TOTAL BLOCKS REQUIRED FOR
HIDDEN LINE WORK.
PQ(2)=MAXIMUM NUMBER OF TOTAL BLOCKS EVER USED AT ONE TIME.

PQ(3)=CURRENT NUMBER OF TOTAL BLOCKS AT A GIVEN TIME.
(USED FOR CALCULATING PQ(2).)
PQ(4)=TOTAL NUMBER OF EDGE BLOCKS IN FRAME.

PQ(5)=NUMBER OF EDGE BLOCKS WITH AT LEAST ONE OF THE
CONNECTED POLYGONS DRAWN CLOCKWISE.
PQ(6)=NUMBER OF THOSE EDGE BLOCKS OF PQ(5) WHOSE Y VALUE
OF THE BEGIN PT IS NOT THE SAME AS THE END PT Y VALUE.
PQ(7)=TOTAL NUMBER OF POLYGON BLOCKS IN THE FRAME.

PQ(8)=NUMBER OF POLYGON BLOCKS DRAWN CLOCKWISE.

PQ(9)=POINT DENSITY.

PQ(10)=NUMBER OF INVOLVED SCAN LINES.

PQ(11)=MEMORY REFERENCES FOR SEGMENT CREATOR.

PQ(12)=NANOSECONDS PER MEMORY REFERENCE FOR SEGMENT CREATOR.

PQ(13)=MEMORY REFERENCES FOR DEPTH CALCULATOR.

PQ(14)=NANOSECONDS PER MEMORY REFERENCE FOR DEPTH CALCULATOR.

PQ(15)=MEMORY REF. TOTAL PQ(11),PQ(13)

PQ(16)=NANOSECONDS FOR PQ(15).

Q COUNTERS

Q(1,X)=TOTAL PER FRAME
Q(2,X)=MAXIMUM REQUIRED OF A SCAN LINE
Q(3,X)=AVERAGE OF TOTAL SCAN LINES. ALSO SCRATCH FOR Q(2,X)
Q(4,X)=REQUIRED FOR PRE-FRAME PROCESSING
Q(5,X)=MAXIMUM REQUIRED FOR SCAN PREPARATION PROCESSING
Q(6,X)=NANOSECONDS REQUIRED. ALSO SCRATCH FOR Q(5,X)
Q(7,X)=MAXIMUM REQUIRED FOR SCAN DEPTH PROCESSING
Q(8,X)=AVERAGE OF ACTIVE SCAN LINES. ALSO SCRATCH FOR Q(7,X)
Q(9,X)=TOTAL FOR SCAN PREPARATION PROCESSING
Q(10,X)=TOTAL FOR SCAN DEPTH PROCESSING

Q(X,1)=NUMBER OF SLOPE CALCULATIONS.
Q(X,2)=NUMBER OF INTERCEPT CALCULATIONS.
Q(X,3)=NUMBER OF SAMPLE POINTS STORED FOR NEXT SCAN LINE.
Q(X,4)=SUBDIVISIONS (NOT FROM INTERSECTING CASE).
Q(X,5)=CONFIRMATIONS (EXCLUDED ARE SIMPLE CHECKS).
Q(X,6)=DEPTH SAMPLES REQUIRED.
Q(X,7)=DEPTH SAMPLES OF Q(X,6) NOT SATISFIED BY MAX-MIN TEST.
Q(X,8)=SAMPLE POINTS DELETED.
Q(X,9)=DEPTH CHECKS REQUIRED IN CONFIRMATION ROUTINE.
Q(X,10)=DEPTH SAMPLES OF Q(X,9) NOT SATISFIED BY MAX-MIN TEST.
Q(X,11)=OUTPUT SEGMENTS.
Q(X,12)=INTERCEPT SUBDIVISIONS.
Q(X,13)=INTERCEPT CALCULATIONS.
Q(X,14)=INTERCEPT SUBDIVISION RESOLUTIONS REACHED.
Q(X,15)=READS FROM POLY.
Q(X,16)=WRITES TO POLY.
Q(X,17)=READS FROM EDGE.
Q(X,18)=WRITES TO EDGE.
Q(X,19)=READS FROM SEG
Q(X,20)=WRITES TO SEG
Q(X,21)=READS FROM BUCKX
Q(X,22)=WRITES TO BUCKX
Q(X,23)=READS FROM FREE LIST(GETBLK)
Q(X,24)=WRITES TO FREE LIST(RETBLK)
Q(X,25)=READS FROM BUCKY
Q(X,26)=WRITES TO BUCKY

THE FOLLOWING HAVE NOT BEEN IMPLEMENTED.

Q(X,27)=READS FROM CX
Q(X,28)=READS FROM CZ
Q(X,29)=READS FROM X
Q(X,30)=READS FROM Y
Q(X,31)=READS FROM Z
Q(X,32)=READS FROM FAST TEMPORARY MEMORY.
Q(X,33)=WRITES TO FAST TEMPORARY MEMORY.

PQ(1)= 54

PQ(2)= 24

PQ(3)= 0

PQ(4)= 105

PQ(5)= 74

PQ(6)= 73

PQ(7)= 49

PQ(8)= 23

PQ(9)=104777

PQ(10)= 387

PQ(11)= 1083

PQ(12)= 30778

PQ(13)= 20647

PQ(14)= 1614

PQ(15)= 21730

PQ(16)= 1533

Q(X, 1)=	73	3	0	0	3	456621	0	0	73	0
Q(X, 2)=	4475	23	8	0	0	7448	23	11	0	4475
Q(X, 3)=	3902	16	7	0	0	8542	16	10	0	3902
Q(X, 4)=	124	3	0	0	0	268817	3	0	0	124
Q(X, 5)=	1296	6	2	0	0	25720	6	3	0	1296
Q(X, 6)=	29067	297	56	0	0	1146	297	75	0	29067
Q(X, 7)=	27630	279	53	0	0	1206	279	71	0	27630
Q(X, 8)=	577	4	1	0	0	57770	4	1	0	577
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,11)=	3978	17	7	0	0	8379	17	10	0	3978
Q(X,12)=	16	4	0	0	0	2083333	4	0	0	16
Q(X,13)=	15	2	0	0	0	2222222	2	0	0	15
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	418	9	0	77	9	79744	5	1	240	101
Q(X,16)=	310	8	0	49	8	107526	5	0	190	71
Q(X,17)=	178	3	0	105	3	187265	0	0	73	0
Q(X,18)=	73	0	0	73	0	456621	0	0	0	0
Q(X,19)=	10625	119	20	0	11	3137	119	27	199	10426
Q(X,20)=	10179	83	19	0	13	3274	82	26	160	10019
Q(X,21)=	48	3	0	0	3	694444	0	0	48	0
Q(X,22)=	48	3	0	0	3	694444	0	0	48	0
Q(X,23)=	54	4	0	0	4	617283	2	0	39	15
Q(X,24)=	54	3	0	0	3	617283	2	0	39	15
Q(X,25)=	47	1	0	0	1	709219	0	0	47	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

PQ(1)= 246

PQ(2)= 40

PQ(3)= 0

PQ(4)= 480

PQ(5)= 380

PQ(6)= 380

PQ(7)= 200

PQ(8)= 110

PQ(9)=124747

PQ(10)= 361

PQ(11)= 6159

PQ(12)= 5412

PQ(13)= 36253

PQ(14)= 919

PQ(15)= 42412

PQ(16)= 785

Q(X, 1)=	380	6	0	0	6	87719	0	1	380	0
Q(X, 2)=	7639	39	14	0	0	4363	38	21	0	7639
Q(X, 3)=	6887	30	13	0	0	4840	30	19	0	6887
Q(X, 4)=	347	5	0	0	0	96061	5	0	0	347
Q(X, 5)=	1169	6	2	0	0	28514	6	3	0	1169
Q(X, 6)=	54630	354	106	0	0	610	354	151	0	54630
Q(X, 7)=	49862	289	97	0	0	668	289	138	0	49862
Q(X, 8)=	608	4	1	0	0	54824	4	1	0	608
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,11)=	7107	30	13	0	0	4690	30	19	0	7107
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	1983	20	3	310	18	16809	7	5	1279	394
Q(X,16)=	1628	16	3	200	15	20475	7	4	1039	389
Q(X,17)=	860	6	1	480	6	38759	0	2	380	0
Q(X,18)=	380	0	0	380	0	87719	0	1	0	0
Q(X,19)=	20477	154	39	0	20	1627	138	56	1236	19241
Q(X,20)=	17288	135	33	0	24	1928	117	47	1059	16229
Q(X,21)=	250	5	0	0	5	133333	0	0	250	0
Q(X,22)=	250	5	0	0	5	133333	0	0	250	0
Q(X,23)=	246	6	0	0	6	135501	0	0	246	0
Q(X,24)=	246	5	0	0	5	135501	0	0	246	0
Q(X,25)=	174	1	0	0	1	191570	0	0	174	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

PQ(1)= 108
PQ(2)= 31
PQ(3)= 0
PQ(4)= 210
PQ(5)= 155
PQ(6)= 153
PQ(7)= 100
PQ(8)= 50
PQ(9)= 53570
PQ(10)= 358
PQ(11)= 2211
PQ(12)= 15076
PQ(13)= 19617
PQ(14)= 1699
PQ(15)= 21828
PQ(16)= 1527

Q(X, 1)=	153	5	0	0	5	217864	0	0	153	0
Q(X, 2)=	4685	30	9	0	0	7114	30	13	0	4685
Q(X, 3)=	4342	24	8	0	0	7676	24	12	0	4342
Q(X, 4)=	211	7	0	0	0	157977	7	0	0	211
Q(X, 5)=	1474	9	2	0	0	22614	9	4	0	1474
Q(X, 6)=	24138	216	47	0	0	1380	216	67	0	24138
Q(X, 7)=	22709	208	44	0	0	1467	208	63	0	22709
Q(X, 8)=	646	4	1	0	0	51599	4	1	0	646
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,11)=	4420	24	8	0	0	7541	24	12	0	4420
Q(X,12)=	10	2	0	0	0	3333333	2	0	0	10
Q(X,13)=	37	3	0	0	0	900900	3	0	0	37
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	882	21	1	138	21	37792	10	2	513	231
Q(X,16)=	672	19	1	100	19	49603	10	1	415	157
Q(X,17)=	363	5	0	210	5	91827	0	1	153	0
Q(X,18)=	153	0	0	153	0	217864	0	0	0	0
Q(X,19)=	9697	104	18	0	20	3437	96	27	407	9290
Q(X,20)=	10157	99	19	0	19	3281	92	28	292	9865
Q(X,21)=	103	4	0	0	4	323624	0	0	103	0
Q(X,22)=	103	4	0	0	4	323624	0	0	103	0
Q(X,23)=	108	6	0	0	6	308641	3	0	71	37
Q(X,24)=	108	5	0	0	5	308641	2	0	71	37
Q(X,25)=	83	1	0	0	1	401606	0	0	83	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

```
PQ( 1)=    25
PQ( 2)=    25
PQ( 3)=     0
PQ( 4)=   300
PQ( 5)=   100.
PQ( 6)=    50
PQ( 7)=   150
PQ( 8)=    25
PQ( 9)=131769
PQ(10)=   363
PQ(11)=   550
PQ(12)= 60606
PQ(13)=  8305
PQ(14)=  4013
PQ(15)=  8855
PQ(16)=  3764
```

Q(X, 1)=	50	2	0	0	2	666666	0	0	50	0
Q(X, 2)=	3579	25	6	0	0	9313	25	9	0	3579
Q(X, 3)=	1236	3	2	0	0	26968	3	3	0	1236
Q(X, 4)=	2	2	0	0	0	16666666	2	0	0	2
Q(X, 5)=	874	2	1	0	0	38138	2	2	0	874
Q(X, 6)=	11829	78	23	0	0	2817	79	32	0	11829
Q(X, 7)=	3102	9	6	0	0	10745	9	8	0	3102
Q(X, 8)=	512	1	1	0	0	65104	1	1	0	512
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,11)=	1238	3	2	0	0	26925	3	3	0	1238
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	325	5	0	150	5	102564	1	0	150	25
Q(X,16)=	275	3	0	150	3	121212	1	0	100	25
Q(X,17)=	350	2	0	300	2	95238	0	0	50	0
Q(X,18)=	50	0	0	50	0	666666	0	0	0	0
Q(X,19)=	4021	29	7	0	2	8289	26	11	75	3946
Q(X,20)=	4359	29	8	0	2	7647	27	10	50	4309
Q(X,21)=	25	1	0	0	1	1333333	0	0	25	0
Q(X,22)=	25	1	0	0	1	1333333	0	0	25	0
Q(X,23)=	25	1	0	0	1	1333333	0	0	25	0
Q(X,24)=	25	1	0	0	1	1333333	0	0	25	0
Q(X,25)=	25	1	0	0	1	1333333	0	0	25	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

```
PQ( 1)=   121
PQ( 2)=    46
PQ( 3)=     0
PQ( 4)=   300
PQ( 5)=   225
PQ( 6)=   215
PQ( 7)=   150
PQ( 8)=    75
PQ( 9)=172829
PQ(10)=   493
PQ(11)=  3128
PQ(12)= 10656
PQ(13)= 70532
PQ(14)=   472
PQ(15)= 73660
PQ(16)=   452
```

Q(X, 1)=	215	4	0	0	4	155038	0	0	215	0
Q(X, 2)=	7705	45	15	0	0	4326	45	15	0	7705
Q(X, 3)=	8153	41	15	0	0	4088	41	16	0	8153
Q(X, 4)=	230	4	0	0	0	144927	4	0	0	230
Q(X, 5)=	999	3	1	0	0	33366	3	2	0	999
Q(X, 6)=	1583311584	309	0	0		2101584	321		0158331	
Q(X, 7)=	1469041407	286	0	0		2261407	297		0146904	
Q(X, 8)=	550	3	1	0	0	60606	3	1	0	550
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,11)=	8291	41	16	0	0	4020	41	16	0	8291
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	1116	14	2	225	14	29868	5	2	698	193
Q(X,16)=	893	13	1	150	12	37327	5	1	550	193
Q(X,17)=	515	4	1	300	4	64724	0	1	215	0
Q(X,18)=	215	0	0	215	0	155038	0	0	0	0
Q(X,19)=	54451	577	106	0	13	612	568	110	567	53884
Q(X,20)=	16728	115	32	0	10	1992	107	33	466	16262
Q(X,21)=	141	3	0	0	3	236406	0	0	141	0
Q(X,22)=	141	3	0	0	3	236406	0	0	141	0
Q(X,23)=	121	3	0	0	3	275482	0	0	121	0
Q(X,24)=	121	4	0	0	4	275482	0	0	121	0
Q(X,25)=	108	1	0	0	1	308641	0	0	108	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

PQ(1)= 50

PQ(2)= 24

PQ(3)= 0

PQ(4)= 201

PQ(5)= 125

PQ(6)= 125

PQ(7)= 100

PQ(8)= 50

PQ(9)= 99425

PQ(10)= 456

PQ(11)= 1497

PQ(12)= 22266

PQ(13)= 33395

PQ(14)= 998

PQ(15)= 34892

PQ(16)= 955

Q(X, 1)=	125	4	0	0	4	266666	0	0	125	0
Q(X, 2)=	6214	24	12	0	0	5364	24	13	0	6214
Q(X, 3)=	5582	19	10	0	0	5971	19	12	0	5582
Q(X, 4)=	160	4	0	0	0	208333	4	0	0	160
Q(X, 5)=	2451	11	4	0	0	13599	11	5	0	2451
Q(X, 6)=	48825	246	95	0	0	682	246	107	0	48825
Q(X, 7)=	41409	201	80	0	0	804	201	90	0	41409
Q(X, 8)=	584	3	1	0	0	57077	3	1	0	584
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,11)=	5635	19	11	0	0	5915	19	12	0	5635
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	625	10	1	175	10	53333	3	1	350	100
Q(X,16)=	500	9	0	100	8	66666	3	1	300	100
Q(X,17)=	326	4	0	201	4	102249	0	0	125	0
Q(X,18)=	125	0	0	125	0	266666	0	0	0	0
Q(X,19)=	16916	96	33	0	6	1970	92	37	250	16666
Q(X,20)=	16679	98	32	0	5	1998	94	36	150	16529
Q(X,21)=	75	3	0	0	3	444444	0	0	75	0
Q(X,22)=	75	3	0	0	3	444444	0	0	75	0
Q(X,23)=	50	2	0	0	2	666666	0	0	50	0
Q(X,24)=	50	2	0	0	2	666666	0	0	50	0
Q(X,25)=	72	1	0	0	1	462962	0	0	72	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

```
PQ( 1)=    75
PQ( 2)=    50
PQ( 3)=     0
PQ( 4)=   201
PQ( 5)=   125
PQ( 6)=   125
PQ( 7)=   100
PQ( 8)=    50
PQ( 9)=102906
PQ(10)=   314
PQ(11)=  1668
PQ(12)= 19984
PQ(13)= 67201
PQ(14)=   496
PQ(15)= 68869
PQ(16)=   484
```

Q(X, 1)=	125	5	0	0	5	266666	0	0	125	0
Q(X, 2)=	12795	50	24	0	0	2605	50	40	0	12795
Q(X, 3)=	11975	52	23	0	0	2783	52	38	0	11975
Q(X, 4)=	236	10	0	0	0	141242	10	0	0	236
Q(X, 5)=	4975	26	9	0	0	6700	26	15	0	4975
Q(X, 6)=	94254	540	184	0	0	353	540	300	0	94254
Q(X, 7)=	87864	540	171	0	0	379	540	279	0	87864
Q(X, 8)=	575	4	1	0	0	57971	4	1	0	575
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,11)=	12026	52	23	0	0	2771	52	38	0	12026
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	625	23	1	175	15	53333	10	1	350	100
Q(X,16)=	500	23	0	100	15	66666	10	1	300	100
Q(X,17)=	326	5	0	201	5	102249	0	1	125	0
Q(X,18)=	125	0	0	125	0	266666	0	0	0	0
Q(X,19)=	32516	210	63	0	15	1025	206	103	300	32216
Q(X,20)=	35035	168	68	0	12	951	165	111	250	34785
Q(X,21)=	75	4	0	0	4	444444	0	0	75	0
Q(X,22)=	75	4	0	0	4	444444	0	0	75	0
Q(X,23)=	75	4	0	0	4	444444	0	0	75	0
Q(X,24)=	75	5	0	0	5	444444	0	0	75	0
Q(X,25)=	43	1	0	0	1	775193	0	0	43	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

```
PQ( 1)=    66
PQ( 2)=    18
PQ( 3)=     0
PQ( 4)=   308
PQ( 5)=   150
PQ( 6)=   147
PQ( 7)=   148
PQ( 8)=    62
PQ( 9)=113820
PQ(10)=   376
PQ(11)=  2296
PQ(12)= 14518
PQ(13)= 12088
PQ(14)=  2757
PQ(15)= 14384
PQ(16)=  2317
```

Q(X, 1)=	147	6	0	0	6	226757	0	0	147	0
Q(X, 2)=	3086	16	6	0	0	10801	16	8	0	3086
Q(X, 3)=	2370	11	4	0	0	14064	11	6	0	2370
Q(X, 4)=	77	4	0	0	0	432900	4	0	0	77
Q(X, 5)=	1172	7	2	0	0	28441	7	3	0	1172
Q(X, 6)=	15591	138	30	0	0	2137	138	41	0	15591
Q(X, 7)=	14301	117	27	0	0	2330	117	38	0	14301
Q(X, 8)=	552	3	1	0	0	60386	3	1	0	552
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,11)=	2402	11	4	0	0	13877	11	6	0	2402
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	897	19	1	177	19	37160	7	2	538	182
Q(X,16)=	818	18	1	148	18	40749	7	2	488	182
Q(X,17)=	455	6	0	308	6	73260	0	1	147	0
Q(X,18)=	147	0	0	147	0	226757	0	0	0	0
Q(X,19)=	6004	72	11	0	16	5551	58	15	434	5570
Q(X,20)=	6414	70	12	0	14	5196	58	17	260	6154
Q(X,21)=	123	5	0	0	5	271002	0	0	123	0
Q(X,22)=	123	5	0	0	5	271002	0	0	123	0
Q(X,23)=	66	5	0	0	5	505050	0	0	66	0
Q(X,24)=	66	2	0	0	2	505050	0	0	66	0
Q(X,25)=	51	1	0	0	1	653594	0	0	51	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

PQ(1)= 89

PQ(2)= 24

PQ(3)= 0

PQ(4)= 308

PQ(5)= 150

PQ(6)= 145

PQ(7)= 148

PQ(8)= 62

PQ(9)=106613

PQ(10)= 363

PQ(11)= 2277

PQ(12)= 14639

PQ(13)= 15268

PQ(14)= 2183

PQ(15)= 17545

PQ(16)= 1899

Q(X, 1)=	145	5	0	0	5	229885	0	0	145	0
Q(X, 2)=	3465	22	6	0	0	9620	22	9	0	3465
Q(X, 3)=	3110	18	6	0	0	10718	18	8	0	3110
Q(X, 4)=	106	4	0	0	0	314465	4	0	0	106
Q(X, 5)=	1199	6	2	0	0	27800	6	3	0	1199
Q(X, 6)=	20013	183	39	0	0	1665	183	55	0	20013
Q(X, 7)=	18817	165	36	0	0	1771	165	51	0	18817
Q(X, 8)=	575	3	1	0	0	57971	3	1	0	575
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,11)=	3176	18	6	0	0	10495	18	8	0	3176
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	22	2	0	0	0	1515151	2	0	0	22
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	931	18	1	177	17	35803	8	2	531	223
Q(X,16)=	810	17	1	148	16	41152	6	2	483	179
Q(X,17)=	453	5	0	308	5	73583	0	1	145	0
Q(X,18)=	145	0	0	145	0	229885	0	0	0	0
Q(X,19)=	8004	98	15	0	14	4164	84	22	431	7573
Q(X,20)=	7511	77	14	0	10	4437	69	20	262	7249
Q(X,21)=	121	4	0	0	4	275482	0	0	121	0
Q(X,22)=	121	4	0	0	4	275482	0	0	121	0
Q(X,23)=	89	3	0	0	3	374531	2	0	67	22
Q(X,24)=	89	4	0	0	4	374531	2	0	67	22
Q(X,25)=	49	1	0	0	1	680272	0	0	49	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

PQ(1)=NUMBER OF TOTAL BLOCKS REQUIRED FOR
HIDDEN LINE WORK.
PQ(2)=MAXIMUM NUMBER OF TOTAL BLOCKS EVER USED AT ONE TIME.

PQ(3)=CURRENT NUMBER OF TOTAL BLOCKS AT A GIVEN TIME.
(USED FOR CALCULATING PQ(2).)
PQ(4)=TOTAL NUMBER OF EDGE BLOCKS IN FRAME.

PQ(5)=NUMBER OF EDGE BLOCKS WITH AT LEAST ONE OF THE
CONNECTED POLYGONS DRAWN CLOCKWISE.
PQ(6)=NUMBER OF T.OSE EDGE BLOCKS OF PQ(5) WHOSE Y VALUE
OF THE BEGIN PT IS NOT THE SAME AS THE END PT Y VALUE.
PQ(7)=TOTAL NUMBER OF POLYGON BLOCKS IN THE FRAME.

PQ(8)=NUMBER OF POLYGON BLOCKS DRAWN CLOCKWISE.

PQ(9)=POINT DENSITY.

PQ(10)=NUMBER OF INVOLVED SCAN LINES.

PQ(11)=MEMORY REFERENCES FOR SEGMENT CREATOR.

PQ(12)=NANOSECONDS PER MEMORY REFERENCE FOR SEGMENT CREATOR.

PQ(13)=MEMORY REFERENCES FOR DEPTH CALCULATOR.

PQ(14)=NANOSECONDS PER MEMORY REFERENCE FOR DEPTH CALCULATOR.

PQ(15)=MEMORY REF. TOTAL PQ(11),PQ(13)

PQ(16)=NANOSECONDS FOR PQ(15).

ADDS(I)=NUMBER OF TIMES THE DEPTH TEST WAS SATISFIED IN.

Q COUNTERS

Q(1,X)=TOTAL PER FRAME
Q(2,X)=MAXIMUM REQUIRED OF A SCAN LINE
Q(3,X)=AVERAGE OF TOTAL SCAN LINES. ALSO SCRATCH FOR Q(2,X)
Q(4,X)=REQUIRED FOR PRE-FRAME PROCESSING
Q(5,X)=MAXIMUM REQUIRED FOR SCAN PREPARATION PROCESSING
Q(6,X)=NANOSECONDS REQUIRED. ALSO SCRATCH FOR Q(5,X)
Q(7,X)=MAXIMUM REQUIRED FOR SCAN DEPTH PROCESSING
Q(8,X)=AVERAGE OF ACTIVE SCAN LINES. ALSO SCRATCH FOR Q(7,X)
Q(9,X)=TOTAL FOR SCAN PREPARATION PROCESSING
Q(10,X)=TOTAL FOR SCAN DEPTH PROCESSING

Q(X,1)=NUMBER OF SLOPE CALCULATIONS.
Q(X,2)=NUMBER OF INTERCEPT CALCULATIONS.
Q(X,3)=NUMBER OF SAMPLE POINTS STORED FOR NEXT SCAN LINE.
Q(X,4)=SUBDIVISIONS (NOT FROM INTERSECTING CASE).
Q(X,5)=
Q(X,6)=DEPTH SAMPLES REQUIRED.
Q(X,7)=
Q(X,8)=SAMPLE POINTS DELETED.
Q(X,9)=
Q(X,10)=OUTPUT SEGMENTS.
Q(X,11)=INTERCEPT CALCULATIONS.
Q(X,12)=INTERCEPT SUBDIVISIONS.
Q(X,13)=OVERHEAD PIPELINE TIME.
Q(X,14)=TIME WAITING FOR CLIPPER.
Q(X,15)=READS FROM POLY.
Q(X,16)=WRITES TO POLY.
Q(X,17)=READS FROM EDGE.
Q(X,18)=WRITES TO EDGE.
Q(X,19)=READS FROM SEG
Q(X,20)=WRITES TO SEG
Q(X,21)=READS FROM BUCKX
Q(X,22)=WRITES TO BUCKX
Q(X,23)=READS FROM FREE LIST(GETBLK)
Q(X,24)=WRITES TO FREE LIST(RETBLK)
Q(X,25)=READS FROM BUCKY
Q(X,26)=USED FOR SHADER

PENETRATION OBJECT VSG5

ADDS

(1)=	1428	(2)=	4318	(3)=	1963	(4)=	879	(5)=	713
(6)=	376	(7)=	145	(8)=	229	(9)=	486	(10)=	106
(11)=	143	(12)=	0	(13)=	0	(14)=	0	(15)=	0
(16)=	0	(17)=	0	(18)=	0	(19)=	0	(20)=	23

PQ(1)= 54

PQ(2)= 23

PQ(3)= 0

PQ(4)= 105

PQ(5)= 74

PQ(6)= 73

PQ(7)= 49

PQ(8)= 23

PQ(9)=104975

PQ(10)= 387

PQ(11)= 1083

PQ(12)= 30778

PQ(13)= 33607

PQ(14)= 991

PQ(15)= 34690

PQ(16)= 960

Q(X, 1)=	73	3	0	0	3	456621	0	0	73	0
Q(X, 2)=	4408	20	8	0	0	7562	20	11	0	4408
Q(X, 3)=	2581	13	5	0	0	12914	13	6	0	2581
Q(X, 4)=	231	12	0	0	0	144300	12	0	0	231
Q(X, 5)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 6)=	9974	194	19	0	0	3342	194	25	0	9974
Q(X, 7)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 8)=	545	11	1	0	0	61162	11	1	0	545
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	3848	16	7	0	0	8662	16	9	0	3848
Q(X,11)=	23	2	0	0	0	1449275	2	0	0	23
Q(X,12)=	17	3	0	0	0	1960784	3	0	0	17
Q(X,13)=	9260	55	18	0	0	3599	55	23	0	9260
Q(X,14)=	3310	142	6	0	0	10070	142	8	0	3310
Q(X,15)=	388	9	0	77	9	85910	5	1	240	71
Q(X,16)=	310	8	0	49	8	107526	5	0	190	71
Q(X,17)=	178	3	0	105	3	187265	0	0	73	0
Q(X,18)=	73	0	0	73	0	456621	0	0	0	0
Q(X,19)=	11251	265	21	0	11	2962	260	29	199	11052
Q(X,20)=	9973	187	19	0	13	3342	187	25	160	9813
Q(X,21)=	48	3	0	0	3	694444	0	0	48	0
Q(X,22)=	48	3	0	0	3	694444	0	0	48	0
Q(X,23)=	54	4	0	0	4	617283	2	0	39	15
Q(X,24)=	54	3	0	0	3	617283	2	0	39	15
Q(X,25)=	47	1	0	0	1	709219	0	0	47	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

ADDS

(1)=	1231	(2)=	10494	(3)=	3748	(4)=	1781	(5)=	1530
(6)=	184	(7)=	30	(8)=	26	(9)=	55	(10)=	46
(11)=	0	(12)=	0	(13)=	0	(14)=	0	(15)=	0
(16)=	0	(17)=	0	(18)=	0	(19)=	0	(20)=	0

PQ(1)= 246

PQ(2)= 40

PQ(3)= 0

PQ(4)= 480

PQ(5)= 380

PQ(6)= 380

PQ(7)= 200

PQ(8)= 110

PQ(9)=124868

PQ(10)= 361

PQ(11)= 6159

PQ(12)= 5412

PQ(13)= 58988

PQ(14)= 565

PQ(15)= 65147

PQ(16)= 511

Q(X, 1)=	380	6	0	0	6	87719	0	1	380	0
Q(X, 2)=	7639	38	14	0	0	4363	38	21	0	7639
Q(X, 3)=	5771	27	11	0	0	5776	27	15	0	5771
Q(X, 4)=	374	12	0	0	0	89126	12	1	0	374
Q(X, 5)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 6)=	18327	198	35	0	0	1818	198	50	0	18327
Q(X, 7)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 8)=	634	12	1	0	0	52576	12	1	0	634
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	7052	30	13	0	0	4726	30	19	0	7052
Q(X,11)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	19254	100	37	0	0	1731	100	53	0	19254
Q(X,14)=	3799	113	7	0	0	8774	113	10	0	3799
Q(X,15)=	1983	20	3	310	18	16809	7	5	1279	394
Q(X,16)=	1628	16	3	200	15	20475	7	4	1039	389
Q(X,17)=	860	6	1	480	6	38759	0	2	380	0
Q(X,18)=	380	0	0	380	0	87719	0	1	0	0
Q(X,19)=	20734	256	40	0	20	1607	253	57	1236	19498
Q(X,20)=	16713	189	32	0	24	1994	188	46	1059	15654
Q(X,21)=	250	5	0	0	5	133333	0	0	250	0
Q(X,22)=	250	5	0	0	5	133333	0	0	250	0
Q(X,23)=	246	6	0	0	6	135501	0	0	246	0
Q(X,24)=	246	5	0	0	5	135501	0	0	246	0
Q(X,25)=	174	1	0	0	1	191570	0	0	174	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

ADDS

(1)=	2174	(2)=	3640	(3)=	1164	(4)=	451	(5)=	418
(6)=	276	(7)=	259	(8)=	243	(9)=	49	(10)=	31
(11)=	11	(12)=	0	(13)=	0	(14)=	0	(15)=	0
(16)=	0	(17)=	0	(18)=	0	(19)=	0	(20)=	61

PQ(1)= 108

PQ(2)= 31

PQ(3)= 0

PQ(4)= 210

PQ(5)= 155

PQ(6)= 153

PQ(7)= 100

PQ(8)= 50

PQ(9)= 53876

PQ(10)= 358

PQ(11)= 2211

PQ(12)= 15076

PQ(13)= 30526

PQ(14)= 1091

PQ(15)= 32737

PQ(16)= 1018

Q(X, 1)=	153	5	0	0	5	217864	0	0	153	0
Q(X, 2)=	4589	29	8	0	0	7263	29	12	0	4589
Q(X, 3)=	2745	17	5	0	0	12143	17	7	0	2745
Q(X, 4)=	312	10	0	0	0	106837	10	0	0	312
Q(X, 5)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 6)=	7822	104	15	0	0	4261	104	21	0	7822
Q(X, 7)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 8)=	743	8	1	0	0	44863	8	2	0	743
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	4195	22	8	0	0	7945	22	11	0	4195
Q(X,11)=	61	4	0	0	0	546448	4	0	0	61
Q(X,12)=	19	3	0	0	0	1754385	3	0	0	19
Q(X,13)=	10020	59	19	0	0	3326	59	27	0	10020
Q(X,14)=	1789	49	3	0	0	18632	49	4	0	1789
Q(X,15)=	808	21	1	138	21	41254	10	2	513	157
Q(X,16)=	672	19	1	100	19	49603	10	1	415	157
Q(X,17)=	363	5	0	210	5	91827	0	1	153	0
Q(X,18)=	153	0	0	153	0	217864	0	0	0	0
Q(X,19)=	9523	153	18	0	20	3500	142	26	407	9116
Q(X,20)=	9505	152	18	0	19	3506	146	26	292	9213
Q(X,21)=	103	4	0	0	4	323624	0	0	103	0
Q(X,22)=	103	4	0	0	4	323624	0	0	103	0
Q(X,23)=	108	6	0	0	6	308641	2	0	71	37
Q(X,24)=	108	6	0	0	5	308641	4	0	71	37
Q(X,25)=	83	1	0	0	1	401606	0	0	83	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

CUBE1 OBJECT VSG5

ADDS

(1)=	363	(2)=	3216	(3)=	0	(4)=	0	(5)=	0
(6)=	0	(7)=	0	(8)=	0	(9)=	0	(10)=	0
(11)=	0	(12)=	0	(13)=	0	(14)=	0	(15)=	0
(16)=	0	(17)=	0	(18)=	0	(19)=	0	(20)=	0

PQ(1)= 25

PQ(2)= 25

PQ(3)= 0

PQ(4)= 300

PQ(5)= 100

PQ(6)= 50

PQ(7)= 150

PQ(8)= 25

PQ(9)=132132

PQ(10)= 363

PQ(11)= 550

PQ(12)= 60606

PQ(13)= 8594

PQ(14)= 3878

PQ(15)= 9144

PQ(16)= 3645

Q(X, 1)=	50	2	0	0	2	666666	0	0	50	0
Q(X, 2)=	3579	25	6	0	0	9313	25	9	0	3579
Q(X, 3)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 4)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 5)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 6)=	3579	25	6	0	0	9313	25	9	0	3579
Q(X, 7)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 8)=	149	1	0	0	0	223713	1	0	0	149
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	1238	3	2	0	0	26925	3	3	0	1238
Q(X,11)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	1386	3	2	0	0	24050	3	3	0	1386
Q(X,14)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,15)=	325	5	0	150	5	102564	1	0	150	25
Q(X,16)=	275	3	0	150	3	121212	1	0	100	25
Q(X,17)=	350	2	0	300	2	95238	0	0	50	0
Q(X,18)=	50	0	0	50	0	666666	0	0	0	0
Q(X,19)=	3654	27	7	0	2	9122	25	10	75	3579
Q(X,20)=	3629	27	7	0	2	9185	25	9	50	3579
Q(X,21)=	25	1	0	0	1	1333333	0	0	25	0
Q(X,22)=	25	1	0	0	1	1333333	0	0	25	0
Q(X,23)=	25	1	0	0	1	1333333	0	0	25	0
Q(X,24)=	25	1	0	0	1	1333333	0	0	25	0
Q(X,25)=	25	1	0	0	1	1333333	0	0	25	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

ADDS

(1)=	976	(2)=	14385	(3)=	39012	(4)=	1	(5)=	0
(6)=	1	(7)=	8	(8)=	33	(9)=	27	(10)=	55
(11)=	17	(12)=	0	(13)=	0	(14)=	0	(15)=	0
(16)=	0	(17)=	0	(18)=	0	(19)=	0	(20)=	0

PQ(1)= 121

PQ(2)= 46

PQ(3)= 0

PQ(4)= 300

PQ(5)= 225

PQ(6)= 215

PQ(7)= 150

PQ(8)= 75

PQ(9)=173177

PQ(10)= 492

PQ(11)= 3128

PQ(12)= 10656

PQ(13)=129438

PQ(14)= 257

PQ(15)=132566

PQ(16)= 251

Q(X, 1)=	215	4	0	0	4	155038	0	0	215	0
Q(X, 2)=	7705	45	15	0	0	4326	45	15	0	7705
Q(X, 3)=	6755	39	13	0	0	4934	39	13	0	6755
Q(X, 4)=	259	11	0	0	0	128700	11	0	0	259
Q(X, 5)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 6)=	53168	554	103	0	0	626	554	108	0	53168
Q(X, 7)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 8)=	244	11	0	0	0	136612	11	0	0	244
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	8221	41	16	0	0	4054	41	16	0	8221
Q(X,11)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	22532	143	44	0	0	1479	143	45	0	22532
Q(X,14)=	36038	435	70	0	0	924	435	73	0	36038
Q(X,15)=	1116	14	2	225	14	29868	5	2	698	193
Q(X,16)=	893	13	1	150	12	37327	5	1	550	193
Q(X,17)=	515	4	1	300	4	64724	0	1	215	0
Q(X,18)=	215	0	0	215	0	155038	0	0	0	0
Q(X,19)=	55341	620	108	0	13	602	614	112	567	54774
Q(X,20)=	16174	119	31	0	10	2060	111	32	466	15708
Q(X,21)=	141	3	0	0	3	236406	0	0	141	0
Q(X,22)=	141	3	0	0	3	236406	0	0	141	0
Q(X,23)=	121	3	0	0	3	275482	0	0	121	0
Q(X,24)=	121	4	0	0	4	275482	0	0	121	0
Q(X,25)=	108	1	0	0	1	308641	0	0	108	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

ADDS

(1)=	660	(2)=	9776	(3)=	4082	(4)=	0	(5)=	0
(6)=	0	(7)=	0	(8)=	1	(9)=	0	(10)=	0
(11)=	0	(12)=	0	(13)=	0	(14)=	0	(15)=	0
(16)=	0	(17)=	0	(18)=	0	(19)=	0	(20)=	0

PQ(1)= 50

PQ(2)= 24

PQ(3)= 0

PQ(4)= 201

PQ(5)= 125

PQ(6)= 125

PQ(7)= 100

PQ(8)= 50

PQ(9)= 99543

PQ(10)= 456

PQ(11)= 1497

PQ(12)= 22266

PQ(13)= 45559

PQ(14)= 731

PQ(15)= 47056

PQ(16)= 708

Q(X, 1)=	125	4	0	0	4	266666	0	0	125	0
Q(X, 2)=	6214	24	12	0	0	5364	24	13	0	6214
Q(X, 3)=	4516	16	8	0	0	7381	16	9	0	4516
Q(X, 4)=	111	3	0	0	0	300300	3	0	0	111
Q(X, 5)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 6)=	14376	71	28	0	0	2318	71	31	0	14376
Q(X, 7)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 8)=	454	3	0	0	0	73421	3	0	0	454
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	5627	19	10	0	0	5923	19	12	0	5627
Q(X,11)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	14791	55	28	0	0	2253	55	32	0	14791
Q(X,14)=	1629	20	3	0	0	20462	20	3	0	1629
Q(X,15)=	625	10	1	175	10	53333	3	1	350	100
Q(X,16)=	500	9	0	100	8	66666	3	1	300	100
Q(X,17)=	326	4	0	201	4	102249	0	0	125	0
Q(X,18)=	125	0	0	125	0	266666	0	0	0	0
Q(X,19)=	14876	81	29	0	6	2240	77	32	250	14626
Q(X,20)=	14463	87	28	0	5	2304	83	31	150	14313
Q(X,21)=	75	3	0	0	3	444444	0	0	75	0
Q(X,22)=	75	3	0	0	3	444444	0	0	75	0
Q(X,23)=	50	2	0	0	2	666666	0	0	50	0
Q(X,24)=	50	2	0	0	2	666666	0	0	50	0
Q(X,25)=	72	1	0	0	1	462962	0	0	72	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

ADDS

(1)=	2122	(2)=	15202	(3)=	7294	(4)=	5411	(5)=	2208
(6)=	500	(7)=	342	(8)=	24	(9)=	0	(10)=	0
(11)=	0	(12)=	0	(13)=	0	(14)=	0	(15)=	0
(16)=	0	(17)=	0	(18)=	0	(19)=	0	(20)=	0

PQ(1)= 75

PQ(2)= 50

PQ(3)= 0

PQ(4)= 201

PQ(5)= 125

PQ(6)= 125

PQ(7)= 100

PQ(8)= 50

PQ(9)=103288

PQ(10)= 314

PQ(11)= 1668

PQ(12)= 19984

PQ(13)=107267

PQ(14)= 310

PQ(15)=108935

PQ(16)= 305

Q(X, 1)=	125	5	0	0	5	266666	0	0	125	0
Q(X, 2)=	12795	50	24	0	0	2605	50	40	0	12795
Q(X, 3)=	10112	49	19	0	0	3296	49	32	0	10112
Q(X, 4)=	907	20	1	0	0	36751	20	2	0	907
Q(X, 5)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 6)=	31923	225	62	0	0	1044	225	101	0	31923
Q(X, 7)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 8)=	1204	20	2	0	0	27685	20	3	0	1204
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	12003	52	23	0	0	2777	52	38	0	12003
Q(X,11)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	34589	176	67	0	0	963	176	110	0	34589
Q(X,14)=	6460	105	12	0	0	5159	105	20	0	6460
Q(X,15)=	625	23	1	175	15	53333	10	1	350	100
Q(X,16)=	500	23	0	100	15	66666	10	1	300	100
Q(X,17)=	326	5	0	201	5	102249	0	1	125	0
Q(X,18)=	125	0	0	125	0	266666	0	0	0	0
Q(X,19)=	34310	266	67	0	15	971	266	109	300	34010
Q(X,20)=	32258	226	63	0	12	1033	226	102	250	32008
Q(X,21)=	75	4	0	0	4	444444	0	0	75	0
Q(X,22)=	75	4	0	0	4	444444	0	0	75	0
Q(X,23)=	75	4	0	0	4	444444	0	0	75	0
Q(X,24)=	75	5	0	0	5	444444	0	0	75	0
Q(X,25)=	43	1	0	0	1	775193	0	0	43	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

ADDS

(1)= 6527	(2)= 825	(3)= 58	(4)= 7	(5)= 1	
(6)= 5	(7)= 4	(8)= Ø	(9)= Ø	(10)= Ø	
(11)= Ø	(12)= Ø	(13)= Ø	[illegible]	[illegible]	
(16)= Ø	(17)= Ø	(18)= Ø	(19)= Ø	(20)= Ø	

PQ(1)= 250

PQ(2)= 31

PQ(3)= Ø

PQ(4)= 308

PQ(5)= 306

PQ(6)= 304

PQ(7)= 192

PQ(8)= 190

PQ(9)= 61282

PQ(10)= 329

PQ(11)= 5605

PQ(12)= 5947

PQ(13)= 35704

PQ(14)= 933

PQ(15)= 41309

PQ(16)= 806

Q(X, 1)=	304	8	0	0	8	109649	0	0	304	0
Q(X, 2)=	6250	28	12	0	0	5333	28	18	0	6250
Q(X, 3)=	5503	25	10	0	0	6057	25	16	0	5503
Q(X, 4)=	275	7	0	0	0	121212	7	0	0	275
Q(X, 5)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 6)=	6790	46	13	0	0	4909	46	20	0	6790
Q(X, 7)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 8)=	204	1	0	0	0	163398	1	0	0	204
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	6929	28	13	0	0	4810	28	21	0	6929
Q(X,11)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	19271	96	37	0	0	1729	96	58	0	19271
Q(X,14)=	6	1	0	0	0	5555555	1	0	0	6
Q(X,15)=	1744	28	3	233	28	19113	10	5	1135	376
Q(X,16)=	1700	28	3	192	28	19607	10	5	1132	376
Q(X,17)=	612	8	1	308	8	54466	0	1	304	0
Q(X,18)=	304	0	0	304	0	109649	0	0	0	0
Q(X,19)=	8764	93	17	0	28	3803	68	26	1062	7702
Q(X,20)=	8779	97	17	0	28	3796	75	26	806	7973
Q(X,21)=	283	8	0	0	8	117785	0	0	283	0
Q(X,22)=	283	8	0	0	8	117785	0	0	283	0
Q(X,23)=	250	9	0	0	9	133333	0	0	250	0
Q(X,24)=	250	6	0	0	6	133333	0	0	250	0
Q(X,25)=	100	1	0	0	1	333333	0	0	100	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

ADDS

(1)=	583	(2)=	985	(3)=	949
(4)=	645	(5)=	364	(6)=	40
(7)=	9	(8)=	2	(9)=	0
(10)=	0	(11)=	0	(12)=	0
(13)=	0	(14)=	0	(15)=	0
(16)=	0	(17)=	0	(18)=	0
(19)=	0	(20)=	0		

PQ(1)= 66

PQ(2)= 18

PQ(3)= 0

PQ(4)= 308

PQ(5)= 150

PQ(6)= 147

PQ(7)= 148

PQ(8)= 62

PQ(9)=113910

PQ(10)= 376

PQ(11)= 2296

PQ(12)= 14518

PQ(13)= 11162

PQ(14)= 2986

PQ(15)= 13458

PQ(16)= 2476

Q(X, 1)=	147	6	0	0	6	226757	0	0	147	0
Q(X, 2)=	3086	16	6	0	0	10801	16	8	0	3086
Q(X, 3)=	670	3	1	0	0	49751	3	1	0	670
Q(X, 4)=	22	10	0	0	0	1515151	10	0	0	22
Q(X, 5)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 6)=	3423	128	6	0	0	9738	128	9	0	3423
Q(X, 7)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 8)=	439	10	0	0	0	75930	10	1	0	439
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	1652	5	3	0	0	20177	5	4	0	1652
Q(X,11)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	2928	41	5	0	0	11384	41	7	0	2928
Q(X,14)=	461	44	0	0	0	72306	44	1	0	461
Q(X,15)=	897	19	1	177	19	37160	7	2	538	182
Q(X,16)=	818	18	1	148	18	40749	7	2	488	182
Q(X,17)=	455	6	0	308	6	73260	0	1	147	0
Q(X,18)=	147	0	0	147	0	226757	0	0	0	0
Q(X,19)=	4033	205	7	0	16	8265	205	10	434	3599
Q(X,20)=	4070	206	7	0	14	8190	206	10	260	3810
Q(X,21)=	123	5	0	0	5	271002	0	0	123	0
Q(X,22)=	123	5	0	0	5	271002	0	0	123	0
Q(X,23)=	66	5	0	0	5	505050	0	0	66	0
Q(X,24)=	66	2	0	0	2	505050	0	0	66	0
Q(X,25)=	51	1	0	0	1	653594	0	0	51	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

ADDS

(1)=	594	(2)=	1991	(3)=	667	(4)=	549	(5)=	341
(6)=	212	(7)=	266	(8)=	263	(9)=	395	(10)=	435
(11)=	0	(12)=	0	(13)=	0	(14)=	0	(15)=	0
(16)=	0	(17)=	0	(18)=	0	(19)=	0	(20)=	41

PQ(1)= 90

PQ(2)= 22

PQ(3)= 0

PQ(4)= 308

PQ(5)= 150

PQ(6)= 145

PQ(7)= 148

PQ(8)= 62

PQ(9)=106696

PQ(10)= 363

PQ(11)= 2277

PQ(12)= 14639

PQ(13)= 18324

PQ(14)= 1819

PQ(15)= 20601

PQ(16)= 1618

Q(X, 1)=	145	5	0	0	5	229885	0	0	145	0
Q(X, 2)=	3339	19	6	0	0	9983	19	9	0	3339
Q(X, 3)=	1364	12	2	0	0	24437	12	3	0	1364
Q(X, 4)=	102	7	0	0	0	326797	7	0	0	102
Q(X, 5)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 6)=	5082	61	9	0	0	6559	61	14	0	5082
Q(X, 7)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 8)=	521	5	1	0	0	63979	5	1	0	521
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	2383	15	4	0	0	13987	15	6	0	2383
Q(X,11)=	41	2	0	0	0	813008	2	0	0	41
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	4133	28	8	0	0	8065	28	11	0	4133
Q(X,14)=	2049	31	4	0	0	16268	31	5	0	2049
Q(X,15)=	887	17	1	177	17	37579	6	2	531	179
Q(X,16)=	810	17	1	148	16	41152	6	2	483	179
Q(X,17)=	453	5	0	308	5	73583	0	1	145	0
Q(X,18)=	145	0	0	145	0	229885	0	0	0	0
Q(X,19)=	6308	108	12	0	14	5284	108	17	431	5877
Q(X,20)=	6123	95	11	0	10	5443	95	16	262	5861
Q(X,21)=	121	4	0	0	4	275482	0	0	121	0
Q(X,22)=	121	4	0	0	4	275482	0	0	121	0
Q(X,23)=	90	3	0	0	3	370370	2	0	67	23
Q(X,24)=	90	4	0	0	4	370370	2	0	67	23
Q(X,25)=	49	1	0	0	1	680272	0	0	49	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

PQ(1)=NUMBER OF TOTAL BLOCKS REQUIRED FOR
HIDDEN LINE WORK.
PQ(2)=MAXIMUM NUMBER OF TOTAL BLOCKS EVER USED AT ONE TIME.

PQ(3)=CURRENT NUMBER OF TOTAL BLOCKS AT A GIVEN TIME.
(USED FOR CALCULATING PQ(2).)
PQ(4)=TOTAL NUMBER OF EDGE BLOCKS IN FRAME.

PQ(5)=NUMBER OF EDGE BLOCKS WITH AT LEAST ONE OF THE
CONNECTED POLYGONS DRAWN CLOCKWISE.
PQ(6)=NUMBER OF THOSE EDGE BLOCKS OF PQ(5) WHOSE Y VALUE
OF THE BEGIN PT IS NOT THE SAME AS THE END PT Y VALUE.
PQ(7)=TOTAL NUMBER OF POLYGON BLOCKS IN THE FRAME.

PQ(8)=NUMBER OF POLYGON BLOCKS DRAWN CLOCKWISE.

PQ(9)=POINT DENSITY.

PQ(10)=NUMBER OF INVOLVED SCAN LINES.

PQ(11)=MEMORY REFERENCES FOR SEGMENT CREATOR.

PQ(12)=NANOSECONDS PER MEMORY REFERENCE FOR SEGMENT CREATOR.

PQ(13)=MEMORY REFERENCES FOR DEPTH CALCULATOR.

PQ(14)=NANOSECONDS PER MEMORY REFERENCE FOR DEPTH CALCULATOR.

PQ(15)=MEMORY REF. TOTAL PQ(11),PQ(13)

PQ(16)=NANOSECONDS FOR PQ(15).

ADDS(I)=NUMBER OF TIMES THE DEPTH TEST WAS SATISFIED IN.

```
Q COUNTERS
Q(1,X)=TOTAL PER FRAME
Q(2,X)=MAXIMUM REQUIRED OF A SCAN LINE
Q(3,X)=AVERAGE OF TOTAL SCAN LINES. ALSO SCRATCH FOR Q(2,X)
Q(4,X)=REQUIRED FOR PRE-FRAME PROCESSING
Q(5,X)=MAXIMUM REQUIRED FOR SCAN PREPARATION PROCESSING
Q(6,X)=NANOSECONDS REQUIRED. ALSO SCRATCH FOR Q(5,X)
Q(7,X)=MAXIMUM REQUIRED FOR SCAN DEPTH PROCESSING
Q(8,X)=AVERAGE OF ACTIVE SCAN LINES. ALSO SCRATCH FOR Q(7,X)
Q(9,X)=TOTAL FOR SCAN PREPARATION PROCESSING
Q(10,X)=TOTAL FOR SCAN DEPTH PROCESSING


Q(X,1)=NUMBER OF SLOPE CALCULATIONS.
Q(X,2)=NUMBER OF INTERCEPT CALCULATIONS.
Q(X,3)=NUMBER OF SAMPLE POINTS STORED FOR NEXT SCAN LINE.
Q(X,4)=SUBDIVISIONS (NOT FROM INTERSECTING CASE).
Q(X,5)=
Q(X,6)=DEPTH SAMPLES REQUIRED.
Q(X,7)=
Q(X,8)=SAMPLE POINTS DELETED.
Q(X,9)=
Q(X,10)=OUTPUT SEGMENTS.
Q(X,11)=INTERCEPT CALCULATIONS.
Q(X,12)=INTERCEPT SUBDIVISIONS.
Q(X,13)=OVERHEAD PIPELINE TIME.
Q(X,14)=TIME WAITING FOR CLIPPER.
Q(X,15)=READS FROM POLY.
Q(X,16)=WRITES TO POLY.
Q(X,17)=READS FROM EDGE.
Q(X,18)=WRITES TO EDGE.
Q(X,19)=READS FROM SEG
Q(X,20)=WRITES TO SEG
Q(X,21)=
Q(X,22)=
Q(X,23)=READS FROM FREE LIST(GETBLK)
Q(X,24)=WRITES TO FREE LIST(RETBLK)
Q(X,25)=READS FROM BUCKY
Q(X,26)=USED FOR SHADER
```

ADDS

(1)=	2033	(2)=	2962	(3)=	2917	(4)=	924	(5)=	469
(6)=	178	(7)=	217	(8)=	482	(9)=	160	(10)=	172
(11)=	3	(12)=	0	(13)=	0	(14)=	0	(15)=	0
(16)=	0	(17)=	0	(18)=	0	(19)=	0	(20)=	27

PQ(1)= 56

PQ(2)= 23

PQ(3)= 0

PQ(4)= 105

PQ(5)= 74

PQ(6)= 73

PQ(7)= 49

PQ(8)= 23

PQ(9)=105065

PQ(10)= 387

PQ(11)= 1263

PQ(12)= 26392

PQ(13)= 32364

PQ(14)= 1029

PQ(15)= 33627

PQ(16)= 991

Q(X, 1)=	73	3	0	0	3	456621	0	0	73	0
Q(X, 2)=	4401	20	8	0	0	7574	20	11	0	4401
Q(X, 3)=	2551	13	4	0	0	13066	13	6	0	2551
Q(X, 4)=	219	12	0	0	0	152207	12	0	0	219
Q(X, 5)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 6)=	9749	194	19	0	0	3419	194	25	0	9749
Q(X, 7)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 8)=	496	11	0	0	0	67204	11	1	0	496
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	3844	16	7	0	0	8671	16	9	0	3844
Q(X,11)=	27	2	0	0	0	1234567	2	0	0	27
Q(X,12)=	17	3	0	0	0	1960784	3	0	0	17
Q(X,13)=	3475	31	6	0	0	9592	31	8	0	3475
Q(X,14)=	5057	238	9	0	0	6591	238	13	0	5057
Q(X,15)=	388	9	0	77	9	85910	5	1	240	71
Q(X,16)=	310	8	0	49	8	107526	5	0	190	71
Q(X,17)=	178	3	0	105	3	187265	0	0	73	0
Q(X,18)=	73	0	0	73	0	456621	0	0	0	0
Q(X,19)=	14238	299	27	0	41	2341	289	36	427	13811
Q(X,20)=	10053	188	19	0	16	3315	188	25	208	9845
Q(X,21)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,22)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,23)=	56	4	0	0	4	595238	2	0	39	17
Q(X,24)=	56	3	0	0	3	595238	2	0	39	17
Q(X,25)=	47	1	0	0	1	709219	0	0	47	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

ADDS

(1)=	5804	(2)=	3813	(3)=	5712	(4)=	2400	(5)=	945
(6)=	206	(7)=	31	(8)=	52	(9)=	46	(10)=	0
(11)=	0	(12)=	0	(13)=	0	(14)=	0	(15)=	0
(16)=	0	(17)=	0	(18)=	0	(19)=	0	(20)=	0

PQ(1)= 246

PQ(2)= 40

PQ(3)= 0

PQ(4)= 480

PQ(5)= 380

PQ(6)= 380

PQ(7)= 200

PQ(8)= 110

PQ(9)=124864

PQ(10)= 361

PQ(11)= 8749

PQ(12)= 3809

PQ(13)= 52919

PQ(14)= 629

PQ(15)= 61668

PQ(16)= 540

Q(X, 1)=	380	6	0	0	6	87719	0	1	380	0
Q(X, 2)=	7639	38	14	0	0	4363	38	21	0	7639
Q(X, 3)=	5771	27	11	0	0	5776	27	15	0	5771
Q(X, 4)=	366	10	0	0	0	91074	10	1	0	366
Q(X, 5)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 6)=	18203	183	35	0	0	1831	183	50	0	18203
Q(X, 7)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 8)=	604	7	1	0	0	55187	7	1	0	604
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	7052	30	13	0	0	4726	30	19	0	7052
Q(X,11)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	6975	42	13	0	0	4778	42	19	0	6975
Q(X,14)=	3516	78	6	0	0	9480	78	9	0	3516
Q(X,15)=	1983	20	3	310	18	16809	7	5	1279	394
Q(X,16)=	1628	16	3	200	15	20475	7	4	1039	389
Q(X,17)=	860	6	1	480	6	38759	0	2	380	0
Q(X,18)=	380	0	0	380	0	87719	0	1	0	0
Q(X,19)=	30148	331	58	0	98	1105	287	83	4076	26072
Q(X,20)=	16882	187	32	0	29	1974	177	46	1309	15573
Q(X,21)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,22)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,23)=	246	6	0	0	6	135501	0	0	246	0
Q(X,24)=	246	5	0	0	5	135501	0	0	246	0
Q(X,25)=	174	1	0	0	1	191570	0	0	174	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

ADDS

(1)=	1790	(2)=	3107	(3)=	2238	(4)=	517	(5)=	367
(6)=	325	(7)=	346	(8)=	108	(9)=	39	(10)=	16
(11)=	3	(12)=	0	(13)=	0	(14)=	0	(15)=	0
(16)=	0	(17)=	0	(18)=	0	(19)=	0	(20)=	62

PQ(1)= 109

PQ(2)= 31

PQ(3)= 0

PQ(4)= 210

PQ(5)= 155

PQ(6)= 153

PQ(7)= 100

PQ(8)= 50

PQ(9)= 53882

PQ(10)= 358

PQ(11)= 2975

PQ(12)= 11204

PQ(13)= 30203

PQ(14)= 1103

PQ(15)= 33178

PQ(16)= 1004

Q(X, 1)=	153	5	0	0	5	217864	0	0	153	0
Q(X, 2)=	4577	29	8	0	0	7282	29	12	0	4577
Q(X, 3)=	2789	18	5	0	0	11951	18	7	0	2789
Q(X, 4)=	335	10	0	0	0	99502	10	0	0	335
Q(X, 5)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 6)=	7931	101	15	0	0	4202	101	22	0	7931
Q(X, 7)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 8)=	766	7	1	0	0	43516	7	2	0	766
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	4193	22	8	0	0	7949	22	11	0	4193
Q(X,11)=	62	4	0	0	0	537634	4	0	0	62
Q(X,12)=	22	3	0	0	0	1515151	3	0	0	22
Q(X,13)=	3917	24	7	0	0	8509	24	10	0	3917
Q(X,14)=	3494	90	6	0	0	9540	90	9	0	3494
Q(X,15)=	808	21	1	138	21	41254	10	2	513	157
Q(X,16)=	672	19	1	100	19	49603	10	1	415	157
Q(X,17)=	363	5	0	210	5	91827	0	1	153	0
Q(X,18)=	153	0	0	153	0	217864	0	0	0	0
Q(X,19)=	14294	191	27	0	50	2331	168	39	1274	13020
Q(X,20)=	9777	157	19	0	23	3409	150	27	395	9382
Q(X,21)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,22)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,23)=	109	6	0	0	6	305810	2	0	71	38
Q(X,24)=	109	6	0	0	5	305810	3	0	71	38
Q(X,25)=	83	1	0	0	1	401606	0	0	83	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

ADDS

(1)= Ø	(2)= 363	(3)= 3216	(4)= Ø	(5)= Ø	
(6)= Ø	(7)= Ø	(8)= Ø	(9)= Ø	(10)= Ø	
(11)= Ø	(12)= Ø	(13)= Ø	(14)= Ø	(15)= Ø	
(16)= Ø	(17)= Ø	(18)= Ø	(19)= Ø	(20)= Ø	

PQ(1)= 25

PQ(2)= 25

PQ(3)= Ø

PQ(4)= 300

PQ(5)= 100

PQ(6)= 50

PQ(7)= 150

PQ(8)= 25

PQ(9)=132132

PQ(10)= 363

PQ(11)= 850

PQ(12)= 39215

PQ(13)= 7719

PQ(14)= 4318

PQ(15)= 8569

PQ(16)= 3889

Q(X, 1)=	50	2	0	0	2	666666	0	0	50	0
Q(X, 2)=	3579	25	6	0	0	9313	25	9	0	3579
Q(X, 3)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 4)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 5)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 6)=	3579	25	6	0	0	9313	25	9	0	3579
Q(X, 7)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 8)=	149	1	0	0	0	223713	1	0	0	149
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	1238	3	2	0	0	26925	3	3	0	1238
Q(X,11)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	512	1	1	0	0	65104	1	1	0	512
Q(X,14)=	24	1	0	0	0	1388888	1	0	0	24
Q(X,15)=	325	5	0	150	5	102564	1	0	150	25
Q(X,16)=	275	3	0	150	3	121212	1	0	100	25
Q(X,17)=	350	2	0	300	2	95238	0	0	50	0
Q(X,18)=	50	0	0	50	0	666666	0	0	0	0
Q(X,19)=	3979	52	7	0	27	8377	25	10	400	3579
Q(X,20)=	3629	28	7	0	3	9185	25	9	75	3554
Q(X,21)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,22)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,23)=	25	1	0	0	1	1333333	0	0	25	0
Q(X,24)=	25	1	0	0	1	1333333	0	0	25	0
Q(X,25)=	25	1	0	0	1	1333333	0	0	25	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

ADDS

(1)=	7299	(2)=	39921	(3)=	7006	(4)=	0	(5)=	0
(6)=	1	(7)=	1	(8)=	34	(9)=	64	(10)=	17
(11)=	0	(12)=	0	(13)=	0	(14)=	0	(15)=	0
(16)=	0	(17)=	0	(18)=	0	(19)=	0	(20)=	0

PQ(1)= 121

PQ(2)= 46

PQ(3)= 0

PQ(4)= 300

PQ(5)= 225

PQ(6)= 215

PQ(7)= 150

PQ(8)= 75

PQ(9)=173177

PQ(10)= 492

PQ(11)= 5058

PQ(12)= 6590

PQ(13)= 86760

PQ(14)= 384

PQ(15)= 91818

PQ(16)= 363

Q(X, 1)=	215	4	0	0	4	155038	0	0	215	0
Q(X, 2)=	7705	45	15	0	0	4326	45	15	0	7705
Q(X, 3)=	6755	39	13	0	0	4934	39	13	0	6755
Q(X, 4)=	254	12	0	0	0	131233	12	0	0	254
Q(X, 5)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 6)=	53049	528	103	0	0	628	528	107	0	53049
Q(X, 7)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 8)=	227	11	0	0	0	146842	11	0	0	227
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	8221	41	16	0	0	4054	41	16	0	8221
Q(X,11)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	7705	51	15	0	0	4326	51	15	0	7705
Q(X,14)=	969	196	1	0	0	34399	196	1	0	969
Q(X,15)=	1116	14	2	225	14	29868	5	2	698	193
Q(X,16)=	893	13	1	150	12	37327	5	1	550	193
Q(X,17)=	515	4	1	300	4	64724	0	1	215	0
Q(X,18)=	215	0	0	215	0	155038	0	0	0	0
Q(X,19)=	64663	663	126	0	92	515	616	131	2638	62025
Q(X,20)=	16282	118	31	0	13	2047	108	33	607	15675
Q(X,21)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,22)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,23)=	121	3	0	0	3	275482	0	0	121	0
Q(X,24)=	121	4	0	0	4	275482	0	0	121	0
Q(X,25)=	108	1	0	0	1	308641	0	0	108	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

ADDS

(1)=	4111	(2)=	4652	(3)=	5604	(4)=	0	(5)=	0
(6)=	0	(7)=	0	(8)=	0	(9)=	0	(10)=	0
(11)=	0	(12)=	0	(13)=	0	(14)=	0	(15)=	0
(16)=	0	(17)=	0	(18)=	0	(19)=	0	(20)=	0

PQ(1)= 50

PQ(2)= 24

PQ(3)= 0

PQ(4)= 201

PQ(5)= 125

PQ(6)= 125

PQ(7)= 100

PQ(8)= 50

PQ(9)= 99531

PQ(10)= 456

PQ(11)= 2162

PQ(12)= 15417

PQ(13)= 38588

PQ(14)= 863

PQ(15)= 40750

PQ(16)= 817

Q(X, 1)=	125	4	0	0	4	266666	0	0	125	0
Q(X, 2)=	6214	24	12	0	0	5364	24	13	0	6214
Q(X, 3)=	4530	16	8	0	0	7358	16	9	0	4530
Q(X, 4)=	89	3	0	0	0	374531	3	0	0	89
Q(X, 5)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 6)=	14248	71	27	0	0	2339	71	31	0	14248
Q(X, 7)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 8)=	439	3	0	0	0	75930	3	0	0	439
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	5630	19	10	0	0	5920	19	12	0	5630
Q(X,11)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	5190	20	10	0	0	6422	20	11	0	5190
Q(X,14)=	99	4	0	0	0	336700	4	0	0	99
Q(X,15)=	625	10	1	175	10	53333	3	1	350	100
Q(X,16)=	500	9	0	100	8	66666	3	1	300	100
Q(X,17)=	326	4	0	201	4	102249	0	0	125	0
Q(X,18)=	125	0	0	125	0	266666	0	0	0	0
Q(X,19)=	19846	143	38	0	51	1679	92	43	990	18856
Q(X,20)=	14468	91	28	0	8	2303	85	31	225	14243
Q(X,21)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,22)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,23)=	50	2	0	0	2	666666	0	0	50	0
Q(X,24)=	50	2	0	0	2	666666	0	0	50	0
Q(X,25)=	72	1	0	0	1	462962	0	0	72	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

ADDS

(1)=	10402	(2)=	6309	(3)=	8256	(4)=	5394	(5)=	2263
(6)=	415	(7)=	388	(8)=	7	(9)=	0	(10)=	0
(11)=	0	(12)=	0	(13)=	0	(14)=	0	(15)=	0
(16)=	0	(17)=	0	(18)=	0	(19)=	0	(20)=	0

PQ(1)= 75

PQ(2)= 50

PQ(3)= 0

PQ(4)= 201

PQ(5)= 125

PQ(6)= 125

PQ(7)= 100

PQ(8)= 50

PQ(9)=103454

PQ(10)= 314

PQ(11)= 2274

PQ(12)= 14658

PQ(13)= 97403

PQ(14)= 342

PQ(15)= 99677

PQ(16)= 334

Q(X, 1)=	125	5	0	0	5	266666	0	0	125	0
Q(X, 2)=	12795	50	24	0	0	2605	50	40	0	12795
Q(X, 3)=	9967	49	19	0	0	3344	49	31	0	9967
Q(X, 4)=	948	20	1	0	0	35161	20	3	0	948
Q(X, 5)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 6)=	32206	225	62	0	0	1035	225	102	0	32206
Q(X, 7)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 8)=	1063	19	2	0	0	31357	19	3	0	1063
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	12023	52	23	0	0	2772	52	38	0	12023
Q(X,11)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	12290	73	24	0	0	2712	73	39	0	12290
Q(X,14)=	6457	74	12	0	0	5162	74	20	0	6457
Q(X,15)=	625	23	1	175	15	53333	10	1	350	100
Q(X,16)=	500	23	0	100	15	66666	10	1	300	100
Q(X,17)=	326	5	0	201	5	102249	0	1	125	0
Q(X,18)=	125	0	0	125	0	266666	0	0	0	0
Q(X,19)=	47199	338	92	0	102	706	338	150	981	46218
Q(X,20)=	32563	224	63	0	14	1023	224	103	325	32238
Q(X,21)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,22)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,23)=	75	4	0	0	4	444444	0	0	75	0
Q(X,24)=	75	5	0	0	5	444444	0	0	75	0
Q(X,25)=	43	1	0	0	1	775193	0	0	43	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

ADDS

(1)=	856	(2)=	6581	(3)=	32	(4)=	4	(5)=	3
(6)=	4	(7)=	1	(8)=	0	(9)=	0	(10)=	0
(11)=	0	(12)=	0	(13)=	0	(14)=	0	(15)=	0
(16)=	0	(17)=	0	(18)=	0	(19)=	0	(20)=	0

PQ(1)= 250

PQ(2)= 31

PQ(3)= 0

PQ(4)= 308

PQ(5)= 306

PQ(6)= 304

PQ(7)= 192

PQ(8)= 190

PQ(9)= 61282

PQ(10)= 329

PQ(11)= 8584

PQ(12)= 3883

PQ(13)= 28826

PQ(14)= 1156

PQ(15)= 37410

PQ(16)= 891

Q(X, 1)=	304	8	0	0	8	109649	0	0	304	0
Q(X, 2)=	6250	28	12	0	0	5333	28	18	0	6250
Q(X, 3)=	5509	25	10	0	0	6050	25	16	0	5509
Q(X, 4)=	281	7	0	0	0	118623	7	0	0	281
Q(X, 5)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 6)=	6820	47	13	0	0	4887	47	20	0	6820
Q(X, 7)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 8)=	204	1	0	0	0	163398	1	0	0	204
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	6935	28	13	0	0	4806	28	21	0	6935
Q(X,11)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	6508	32	12	0	0	5121	32	19	0	6508
Q(X,14)=	9	2	0	0	0	3703703	2	0	0	9
Q(X,15)=	1744	28	3	233	28	19113	10	5	1135	376
Q(X,16)=	1700	28	3	192	28	19607	10	5	1132	376
Q(X,17)=	612	8	1	308	8	54466	0	1	304	0
Q(X,18)=	304	0	0	304	0	109649	0	0	0	0
Q(X,19)=	18063	206	35	0	118	1845	99	54	4324	13739
Q(X,20)=	8907	104	17	0	36	3742	80	27	1089	7818
Q(X,21)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,22)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,23)=	250	9	0	0	9	133333	0	0	250	0
Q(X,24)=	250	6	0	0	6	133333	0	0	250	0
Q(X,25)=	100	1	0	0	1	333333	0	0	100	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

ADDS

(1)=	249	(2)=	600	(3)=	663	(4)=	892	(5)=	593
(6)=	356	(7)=	21	(8)=	9	(9)=	0	(10)=	0
(11)=	0	(12)=	0	(13)=	0	(14)=	0	(15)=	0
(16)=	0	(17)=	0	(18)=	0	(19)=	0	(20)=	0

PQ(1)= 66

PQ(2)= 18

PQ(3)= 0

PQ(4)= 308

PQ(5)= 150

PQ(6)= 147

PQ(7)= 148

PQ(8)= 62

PQ(9)=114088

PQ(10)= 376

PQ(11)= 2958

PQ(12)= 11268

PQ(13)= 9954

PQ(14)= 3348

PQ(15)= 12912

PQ(16)= 2581

Q(X, 1)=	147	6	0	0	6	226757	0	0	147	0
Q(X, 2)=	3086	16	6	0	0	10801	16	8	0	3086
Q(X, 3)=	492	3	0	0	0	67750	3	1	0	492
Q(X, 4)=	9	4	0	0	0	3703703	4	0	0	9
Q(X, 5)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 6)=	3295	50	6	0	0	10116	50	8	0	3295
Q(X, 7)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 8)=	249	3	0	0	0	133868	3	0	0	249
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	1652	5	3	0	0	20177	5	4	0	1652
Q(X,11)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	1022	9	1	0	0	32615	9	2	0	1022
Q(X,14)=	1131	15	2	0	0	29472	15	3	0	1131
Q(X,15)=	897	19	1	177	19	37160	7	2	538	182
Q(X,16)=	818	18	1	148	18	40749	7	2	488	182
Q(X,17)=	455	6	0	308	6	73260	0	1	147	0
Q(X,18)=	147	0	0	147	0	226757	0	0	0	0
Q(X,19)=	5142	107	10	0	56	6482	81	13	1219	3923
Q(X,20)=	3897	99	7	0	19	8553	89	10	383	3514
Q(X,21)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,22)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,23)=	66	5	0	0	5	505050	0	0	66	0
Q(X,24)=	66	2	0	0	2	505050	0	0	66	0
Q(X,25)=	51	1	0	0	1	653594	0	0	51	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0

ADDS

(1)=	1216	(2)=	928	(3)=	857	(4)=	399	(5)=	536
(6)=	279	(7)=	207	(8)=	360	(9)=	639	(10)=	249
(11)=	0	(12)=	0	(13)=	0	(14)=	0	(15)=	0
(16)=	0	(17)=	0	(18)=	0	(19)=	0	(20)=	41

PQ(1)= 90

PQ(2)= 22

PQ(3)= 0

PQ(4)= 308

PQ(5)= 150

PQ(6)= 145

PQ(7)= 148

PQ(8)= 62

PQ(9)=106696

PQ(10)= 363

PQ(11)= 2994

PQ(12)= 11133

PQ(13)= 21405

PQ(14)= 1557

PQ(15)= 24399

PQ(16)= 1366

Q(X, 1)=	145	5	0	0	5	229885	0	0	145	0
Q(X, 2)=	3332	19	6	0	0	10004	19	9	0	3332
Q(X, 3)=	1382	12	2	0	0	24119	12	3	0	1382
Q(X, 4)=	97	7	0	0	0	343642	7	0	0	97
Q(X, 5)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 6)=	5051	61	9	0	0	6599	61	13	0	5051
Q(X, 7)=	0	0	0	0	0	000000000	0	0	0	0
Q(X, 8)=	513	5	1	0	0	64977	5	1	0	513
Q(X, 9)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,10)=	2383	15	4	0	0	13987	15	6	0	2383
Q(X,11)=	41	2	0	0	0	813008	2	0	0	41
Q(X,12)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,13)=	2077	15	4	0	0	16048	15	5	0	2077
Q(X,14)=	5650	90	11	0	0	5899	90	15	0	5650
Q(X,15)=	887	17	1	177	17	37579	6	2	531	179
Q(X,16)=	810	17	1	148	16	41152	6	2	483	179
Q(X,17)=	453	5	0	308	5	73583	0	1	145	0
Q(X,18)=	145	0	0	145	0	229885	0	0	0	0
Q(X,19)=	8789	124	17	0	56	3792	124	24	1269	7520
Q(X,20)=	6137	95	11	0	13	5431	95	16	383	5754
Q(X,21)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,22)=	0	0	0	0	0	000000000	0	0	0	0
Q(X,23)=	90	4	0	0	3	370370	2	0	67	23
Q(X,24)=	90	4	0	0	4	370370	2	0	67	23
Q(X,25)=	49	1	0	0	1	680272	0	0	49	0
Q(X,26)=	0	0	0	0	0	000000000	0	0	0	0